Optimisation et application du codage réseau dans l'architecture des futurs réseaux sans fils

Samih Abdul-Nabi

Thèse

THESE INSA Rennes
sous le sceau de l'Université européenne de Bretagne
pour obtenir le titre de
DOCTEUR DE L'INSA DE RENNES
Spécialité : Electronique et Télécommunications

présentée par
Samih Abdul-Nabi
ECOLE DOCTORALE : *MATISSE*
LABORATOIRE : *IETR*

Centralized and distributed address correlated network protocols

Soutenance prévue le 28.09.2015
devant le jury composé de :

Jean-Pierre CANCES
Professeur à l'ENSIL de Limoges / Rapporteur
Michel TERRE
Professeur au CNAM de Paris / Rapporteur
Marie-Laure BOUCHERET
Professeur à l'ENSEEIHT de Toulouse / Examinateur
Guillaume GELLE
Professeur à l'Université de Reims / Examinateur
Ayman KHALIL
Enseignant-Chercheur à l'Univ. Internationale du Liban / Co-encadrant de thèse
Philippe MARY
Maître de Conférences à l'INSA de Rennes / Co-encadrant de thèse
Jean-François HELARD
Professeur à l'INSA de Rennes / Directeur de thèse

Centralized and distributed address correlated network protocols

Samih Abdul-Nabi

En partenariat avec

Acknowledgment

I would like to extend thanks to many people who so generously contributed, directly and indirectly, to the work presented in this thesis.

First and foremost, I would like to express my special appreciation and thanks to my adviser, Prof. Jean-François HÉLARD for this great opportunity, tremendous academic support, guidance and encouragement.

Likewise, Similar, profound gratitude goes to Dr. Philippe MARY for his supervision, crucial contribution to this thesis, and for the long discussions we had about network coding.

I would like to express my sincere thanks to Dr. Ayman KHALIL who showed me the way to Rennes, for his supervision and the long after hour discussions.

I would like to thank Prof. Guillaume GELLE, Prof. Jean-Pierre CANCES, Prof. Marie-Laure BOUCHERET and Prof. Michel TERRE for accepting to be jury members in my PhD committee.

I would like to show my love to my wife Ilfat, my daughter Sarah and my son Ahmad for their support and encouragement.

Last but not the least ; I would like to thank both INSA and LIU directors for making this thesis possible.

Table of contents

Acronyms

ACK	Acknowledgment
ACNC	Address Correlated Network Coding
ARQ	Automatic Repeat Request
BCSD	Basic Covering Set Discovery
BIP	Binary Integer Programming
CA	Coding Activity
CDMA	Code Division Multiple Access
CFNC	Complex Field Network Coding
CFP	Coding and Forwarding Process
DARPA	Defence Advanced Research Projects Agency
FDMA	Frequency Division Multiple Access
GF	Galois Field
HetNet	Heterogeneous Networks
INC	Institute of network coding
IoT	Internet of Things
IRR	Immediate Retransmission Request
IRTF	Internet Engineering Task Force
ITMANET	Information Theory for Mobile Ad Hoc Networks
LAN	Local Area Network
LCANC	Low Cost Advanced Network Coding
LDRM	Loss Detection and Recovery Mechanism
LNC	Linear Network Coding
MAC	Media Access Control
NACK	Negative Acknowledgment
NAT	Network Address Translation
NC	Network Coding
NCRAVE	Network Coding for Robust Architectures in Volatile Environments
NP-Hard	Non-deterministic Polynomial-time hard
NWCRG	Network Coding Research Group
OPNET	Optimized Network Engineering Tools
OSI	Open Systems Interconnection

P2P	Peer to Peer
PrNC	Partial Network Coding
PNC	Physical layer Network Coding
QoS	Qualities of Service
RCFP	Receiving, Coding and Forwarding Process
RP	Receiving Process
RCNC	Relay Correlated Network Coding
RNC	Random Network Coding
SF	Store and Forward
TDMA	Time Division Multiple Access
TNC	Traditional Network Coding
UDP	User Datagram Protocol
VANET	Vehicular Ad-Hoc Networks
WMN	Wireless Mesh Networks
WSN	Wireless Sensor Network
NCRAVE	Network Coding for Robust Architectures in Volatile Environments

List of figures

List of tables

Résumé en Français

Le codage de réseau (CR) est une nouvelle technique reposant, sur la réalisation par les nœuds du réseau, des fonctions de codage et de décodage des données afin d'améliorer le débit et réduire les retards. En utilisant des algorithmes algébriques, le codage consiste à combiner ensemble les paquets transmis et le décodage consiste à restaurer ces paquets. Cette opération permet de réduire le nombre total de transmissions de paquets pour échanger les données, mais requière des traitements additionnels au niveau des nœuds. Le codage de réseau peut être appliqué au niveau de différentes couches ISO. Toutefois dans ce travail, sa mise en nœuvre est effectuée au niveau de la couche réseau. Dans ce travail de thèse, nous présentons des techniques de codage de réseau s'appuyant sur de nouveaux protocoles permettant d'optimiser l'utilisation de la bande passante, d'améliorer la qualité de service et de réduire l'impact de la perte de paquets dans les réseaux à pertes. Plusieurs défis ont été relevés notamment concernant les fonctions de codage/décodage et tous les mécanismes connexes utilisés pour livrer les paquets échangés entre les nœuds. Des questions comme le cycle de vie des paquets dans le réseau, la cardinalité des messages codés, le nombre total d'octets transmis et la durée du temps de maintien des paquets ont été adressées analytiquement, en s'appuyant sur des théorèmes, qui ont été ensuite confirmés par des simulations. Dans les réseaux à pertes, les méthodes utilisées pour étudier précisément le comportement du réseau conduisent à la proposition de nouveaux mécanismes pour surmonter cette perte et réduire la charge. Dans la première partie de la thèse, un état de l'art des techniques de codage de réseaux est présenté à partir des travaux de Alshwede et al. Les différentes techniques sont détaillées mettant l'accent sur les codages linéaires et binaires. Ces techniques sont décrites en s'appuyant sur différents scénarios pour aider à comprendre les avantages et les inconvénients de chacune d'elles.

Dans la deuxième partie, un nouveau protocole basé sur la corrélation des adresses

(ACNC) est présenté, et deux approches utilisant ce protocole sont introduites ; l'approche centralisée où le décodage se fait aux nœuds d'extrémités et l'approche distribuée où chaque nœud dans le réseau participe au décodage. Le décodage centralisé est élaboré en présentant d'abord ses modèles de décision et le détail du décodage aux nœuds d'extrémités. La cardinalité des messages codés reçus et les exigences de mise en mémoire tampon au niveau des nœuds d'extrémités sont étudiées et les notions d'âge et de maturité sont introduites. On montre que le décodage distribué permet de réduire la charge sur les nœuds d'extrémité ainsi que la mémoire tampon au niveau des nœuds intermédiaires. La perte et le recouvrement avec les techniques de codage de réseau sont examinés pour les deux approches proposées. Pour l'approche centralisée, deux mécanismes pour limiter l'impact de la perte sont présentés. A cet effet, le concept de fermetures et le concept des sous-ensembles couvrants sont introduits. Les recouvrements optimaux afin de trouver l'ensemble optimal de paquets à retransmettre dans le but de décoder tous les paquets reçus sont définis. Pour le décodage distribué, un nouveau mécanisme de fiabilité saut à saut est proposé tirant profit du codage de réseau et permettant de récupérer les paquets perdus sans la mise en œuvre d'un mécanisme d'acquittement.

Abstract

Network coding (NC) is a new technique in which transmitted data is encoded and decoded by the nodes of the network in order to enhance throughput and reduce delays. Using algebraic algorithms, encoding at nodes accumulates various packets in one message and decoding restores these packets. NC requires fewer transmissions to transmit all the data but more processing at the nodes. NC can be applied at any of the ISO layers. However, the focus is mainly on the network layer level. In this work, we introduce novelties to the NC paradigm with the intent of building easy to implement NC protocols in order to improve bandwidth usage, enhance QoS and reduce the impact of losing packets in lossy networks. Several challenges are raised by this thesis concerning details in the coding and decoding processes and all the related mechanisms used to deliver packets between end nodes. Notably, questions like the life cycle of packets in coding environment, cardinality of coded messages, number of bytes overhead transmissions and buffering time duration are inspected, analytically counted, supported by many theorems and then verified through simulations. By studying the packet loss problem, new theorems describing the behavior of the network in that case have been proposed and novel mechanisms to overcome this loss have been provided. In the first part of the thesis, an overview of NC is conducted since triggered by the work of Alshwede et al. NC techniques are then detailed with the focus on linear and binary NC. These techniques are elaborated and embellished with examples extracted from different scenarios to further help understanding the advantages and disadvantages of each of these techniques.

In the second part, a new address correlated NC (ACNC) protocol is presented and two approaches using ACNC protocol are introduced, the centralized approach where decoding is conducted at end nodes and the distributed decoding approach where each node in the network participates in the decoding process. Centralized decoding is elab-

orated by first presenting its decision models and the detailed decoding procedure at end nodes. Moreover, the cardinality of received coded messages and the buffering requirements at end nodes are investigated and the concepts of aging and maturity are introduced. The distributed decoding approach is presented as a solution to reduce the overhead on end nodes by distributing the decoding process and buffering requirements to intermediate nodes.

Loss and recovery in NC are examined for both centralized and distributed approaches. For the centralized decoding approach, two mechanisms to limit the impact of loss are presented. To this effect, the concept of closures and covering sets are introduced and the covering set discovery is conducted on undecodable messages to find the optimized set of packets to request from the sender in order to decode all received packets. For the distributed decoding, a new hop-to-hop reliability mechanism is proposed that takes advantage of the NC itself and depicts loss without the need of an acknowledgement mechanism.

Résumé étendu de la thèse en français

Introduction générale

L'un des principaux défis en matière de communication de réseau est de répondre à la croissante demande de la bande passante afin d'offrir un service d'échange de données, sans compromettre la qualité de service (QoS) exigée, et sans la nécessité de changer l'infrastructure du réseau. Les réseaux sans fil constituent une part de plus en plus importante des infrastructures de communication. La demande en bande passante est ainsi en augmentation continue et le mécanisme traditionnel de stockage et retransmission utilisé pour transmettre des paquets entre les nœuds devient de plus en plus gourmand en bande passante. Face à ce défi, le codage de réseau (CR) où les paquets sont combinés, offre des possibilités extrêmement intéressantes pour réduire l'utilisation de la bande passante tout en répondant aux exigences de communication.

De nombreuses solutions d'accès multiple ont été adoptées pour partager la bande passante entre plusieurs utilisateurs. Ces solutions sont basées sur des techniques d'accès au canal comme l'accès multiple à répartition dans le temps (AMRT), l'accès multiple par répartition en fréquence (AMRF) et l'accès multiple par répartition par le code (AMRC). Ce dernier permet à de multiples terminaux de transmettre en même temps et sur la même bande en partageant la puissance entre les utilisateurs. Le partage de la bande passante a été proposé comme une solution face à la limitation du spectre sans fil [1], et le CR analogique [2] permet des transmissions multiples résistant à la présence d'interférences.

Le CR est une nouvelle technique dans laquelle les données sont codées et décodées par l'intermédiaire des nœuds afin d'améliorer le débit et réduire le retard. En utilisant des algorithmes algébriques, le codage consiste à combiner les paquets et le décodage consiste à restaurer ces paquets. Ces opérations réduisent le nombre de transmissions

nécessaires pour échanger les données, mais requièrent plus de traitement au niveau des nœuds. Le CR peut être appliqué à l'une ou l'autre des couches ISO ; cependant dans ce travail, le CR est appliqué au niveau de la couche réseau.

Dans le cadre de cette thèse, nous présentons des contributions novatrices au CR. Notre objectif est de construire des protocoles pour améliorer l'utilisation de la bande passante, et la qualité de service. A cet effet, nous proposons des solutions concernant le principe de codage et de décodage et des mécanismes pour livrer les paquets échangés entre les nœuds. Notamment, des questions comme le cycle de vie des paquets dans le réseau, la cardinalité des messages codés, le nombre total d'octets transmis et la durée du temps de maintien des paquets ont été adressées, analytiquement obtenus, soutenus par de nombreux théorèmes puis vérifiés par des simulations.

Dans la première partie de la thèse, nous présentons un aperçu du CR depuis les travaux fondateurs d'Alshwede et al. Les différentes techniques sont détaillées mettant l'accent sur le codage linéaire et binaire. Ces techniques sont étudiées à l'aide des exemples extraits de différents scénarios pour comprendre les avantages et les inconvénients de chacune d'elles.

Dans la deuxième partie, nous présentons un nouveau protocole basé sur la corrélation des adresses (ACNC), puis nous introduisons deux approches utilisant ce protocole. Dans la première approche, appelé centralisée, le décodage se limite aux nœuds d'extrémités alors que dans la deuxième approche, appelé distribuée, chaque nœud du réseau participe au décodage. Nous étudions ces deux approches, leurs modèles de décision, les principes de décodage, la cardinalité des messages codés reçus puis nous introduisons les notions d'âge et de maturité. Les résultats indiquent que la méthode de décodage distribué représente une solution qui réduit la charge sur les nœuds d'extrémités en distribuant le processus de décodage et les exigences en mémoire tampon aux nœuds intermédiaires.

Nous traitons aussi la perte et le recouvrement en CR pour les deux approches proposées. Nous présentons deux mécanismes pour limiter l'impact de la perte pour l'approche centralisée. A cet effet, les concepts de fermeture et des ensembles couvrants sont introduits ainsi que la découverte des revêtements optimaux pour trouver l'ensemble optimal de paquets à retransmettre afin de décoder tous les paquets reçus. Pour le décodage distribué, nous étudions un nouveau mécanisme de fiabilité saut à saut qui tire profit du CR et qui permet d'obtenir une transmission sans perte en l'absence d'un mécanisme d'acquittement.

Chapitre 1
Principe du codage de réseau

Au lieu de transmettre simplement les paquets un à un, le CR permet aux nœuds émetteurs de combiner plusieurs paquets en un seul paquet codé qui est ensuite transmis dans le réseau. L'idée de base du CR est illustrée par l'exemple de la Figure 1. Dans cette figure, représentant le célèbre réseau en papillon, le nœud s transmet en multidiffusion aux nœuds t_1 et t_2 les paquets P_1 et P_2. Le nœud 3 reçoit les paquets P_1 et P_2 afin de les transmettre sur le lien 3-4. Traditionnellement, deux utilisations de canal sont nécessaires pour transmettre les deux paquets. Avec le CR, le nœud 3 n'a besoin que d'une seule utilisation de canal en transmettant $P_1 \oplus P_2$ ($\oplus$ étant l'opération au niveau du bit XOR) économisant ainsi une utilisation de canal. En recevant $P_1 \oplus P_2$ les nœuds t_1 et t_2 sont chacun en mesure de restituer les deux paquets P_1 et P_2.

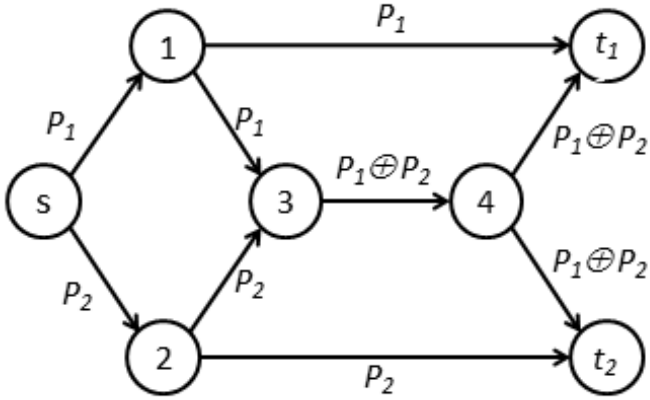


Figure 1 – Exemple de CR : Réseau en papillon

Aperçu historique

La Figure 2 montre le progrès du CR depuis qu'il a été révélé en 2000 par Ahlswede et al. [3] jusqu'à la création du groupe de recherche CR (NWCRG - Network Coding Research Group) en 2013. Dans cette figure, le nombre de citations indiqué par Google Scholar est affiché à côté de chacune des publications qui ont marqué l'évolution du CR.

Après l'initiation par Ahlswede et al. [3], le CR est enrichi en 2003 par le travail de Li et al. [4]. Depuis, il y a eu de nombreuses tentatives vers la mise en œuvre pratique du CR

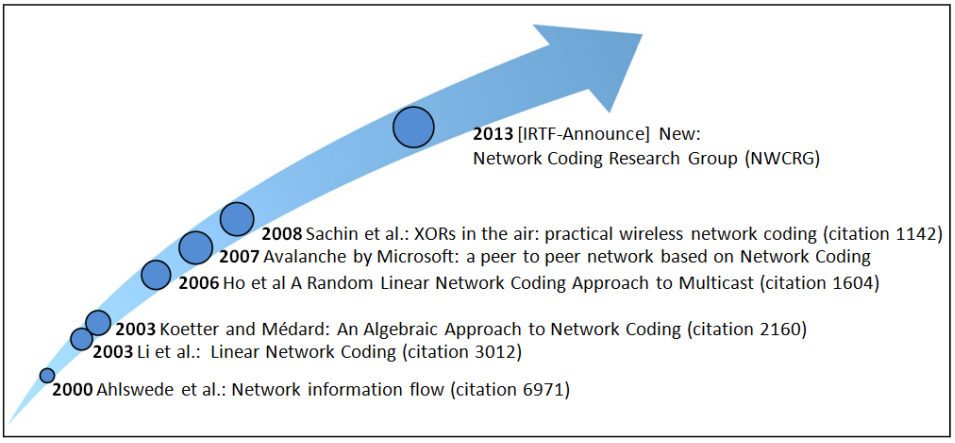


FIGURE 2 – Un coup d'œil sur le codage réseau

dans des applications réelles. En particulier, Koetter et Médard dans [5] ont développé des techniques basées sur des considérations algébriques et montrent équivalence entre ces techniques et le CR.

En 2008, Microsoft a lancé Avalanche, une application point à point (P2P) similaire à BitTorrent (protocole de distribution et de partage de fichiers) mais qui améliore certains de ses inconvénients. Avalanche divise le fichier en petits blocs pour ensuite les distribuer. Toutefois, contrairement à BitTorrent, il utilise le CR sur ces blocs de façon a réduire la complexité et la gestion des blocs. Dans [6], les auteurs ont prouvé, à travers des simulations, que le temps de téléchargement de fichiers est amélioré de 2 à 3 fois par rapport à l'envoi d'informations non codées.

Toujours en 2008, Sachin et al. [7] ont publié un protocole complet basé sur le CR nommé COPE. COPE ajoute une couche de codage, entre la couche MAC et la couche Réseau. Cette nouvelle couche est utilisée pour identifier les possibilités de codage disponibles dans le but de sauver des transmissions. En 2013, le NWCRG est créé dans le but de rechercher des principes et des méthodes de CR qui peuvent être bénéfiques pour la communication sur Internet.

Conclusion

Dans ce chapitre, un aperçu général de CR a été présenté et le domaine de recherche clé en CR a été identifié. En outre, les processus de codage/décodage ont été bien étudiés et l'intérêt du CR a été identifié. Parmi les domaines d'intérêt, cette thèse se concentre

sur l'application du CR afin d'améliorer le débit et réduire le temps d'attente. En fait, au cours des trois dernières années, nous nous sommes intéressés à la définition d'un protocole basé sur la corrélation d'adresses afin de mettre en œuvre pratiquement le codage réseau. Ce concept sera représenté en détail dans le prochain chapitre.

Chapitre 2
Codage de réseau à corrélation d'adresse

Le nouveau CR à corrélation d'adresse (CRCA) est introduit dans ce chapitre où les nœuds intermédiaires (ou relais) ne codent ensemble que des paquets qui vérifient la corrélation d'adresse. La corrélation d'adresse est définie comme suit :

Définition 1. *Deux paquets vérifient la corrélation d'adresse si la destination de l'un des deux paquets est la source de l'autre.*

La contribution de ce chapitre est double :

- d'abord, créer une nouvelle approche de routage pour diriger les messages codés vers diverses destinations.
- deuxièmement, proposer un algorithme basé sur des indicateurs adaptatifs pour prédire la réception des paquets par les destinations afin d'utiliser davantage ces paquets dans des opérations de codage et garantir la décodabilité à la réception.

Le modèle CRCA

Dans cette thèse, nous utilisons un réseau linéaire identique à celui de la Figure 3. Ces réseaux sont formés par un ensemble de nœuds d'extrémités, tels que A et B échangeant des informations à travers un ensemble de nœuds intermédiaires (NI). Dans ces réseaux, les utilisateurs peuvent échanger différents types d'informations, y compris des séquences vidéo, de la messagerie urgente et des fichiers. Cette variété de types de messages nécessite de prioriser des messages afin de maintenir un niveau élevé de QoS. L'accent est mis sur la sélection des paquets reçus et sélectionnés par les relais afin d'être codés et émis dans le réseau.

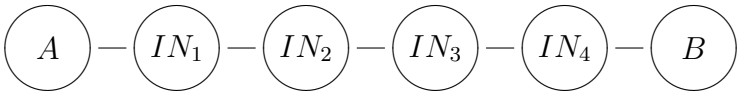


FIGURE 3 – Un réseau lineaire avec 4 nœuds intermédiaires

Quand un paquet i arrive à un nœud intermédiaire NI_j, il est mis en attente dans la file d'attente du nœud et un *timer* t_i, initialement fixé à 0, est associé au paquet i.

La priorité de chaque paquet est indiquée par Pr_i. De plus, on définit $T(Pr_i)$ comme le temps maximal d'attente d'un paquet de priorité Pr_i. Une variable binaire y_i, initialement fixé à 0, est également associée à chaque paquet reçu ; il est mis à 1 lorsque le paquet i est sélectionné pour être codé.

La sélection des paquets à coder et émettre dans le réseau est réalisée en deux phases. Cette sélection est déclenchée quand le temps passé par un paquet dans la file d'attente est expiré. Durant la phase 1, les paquets élites (de priorités élevées) sont sélectionnés pour être codés. Si le nombre de paquets choisis n'est pas suffisant, la phase 2 est utilisée pour ajouter à l'ensemble déjà choisi, des candidats capables d'augmenter la corrélation entre les paquets et ainsi améliorer l'utilisation de la bande passante.

Quand suffisamment de paquets ont été sélectionnés après ces deux phases, des messages codés sont construits et envoyés dans le réseau. Le processus se répète alors avec l'expiration d'un nouveau paquet dans la file d'attente. Les problèmes d'optimisation modélisant les phases 1 et 2 sont présentés par (1) et (2). Un algorithme de faible complexité, est proposé pour la sélection et la création des messages codés dans un délai acceptable.

$$\begin{aligned}
&\min \ \textstyle\sum_{i=1}^{n}(P_w(i)+E_w(i))y_i \\
&\text{Sous contraintes} \\
&\begin{cases}
C1 : \sum_{i=1}^{n} y_i \leq M_d \\
C2 : t_i - (T(P_i) - P - t_d) \leq y_i \times C \ \text{ for } i{=}1 \text{ to } n \\
y_i \in \{0,1\}
\end{cases}
\end{aligned} \tag{1}$$

$$\begin{aligned}
&\max \ \textstyle\sum_{i=1}^{n} y_i \\
&\text{Sous contraintes} \\
&\begin{cases}
C1 : \sum_{i=1}^{n} y_i \leq M_d - |\overline{P_1}| \\
C3 : \sum_{i=1}^{n} \min\,(1, |S_j - D_i| \times y_i y_j) < \sum_{i=1}^{n} y_i \ \forall j \\
C4 : \sum_{i=1}^{n} \min\,(1, |C_j - C_i| \times y_i y_j) < \sum_{i=1}^{n} y_i \ \forall j \\
y_i \in \{0,1\}
\end{cases}
\end{aligned} \tag{2}$$

Une mise en œuvre pratique de CRCA

Le CRCA ajoute une couche de CR entre la couche MAC et la couche réseau. En général, le CRCA génère un message codé à partir de paquets codés ensemble à l'aide de l'opération XOR sur les bits des paquets. Un entête est ajouté à chaque message codé décrivant les paquets codés dans le message et la corrélation entre ces paquets. Le CRCA utilise également un indicateur adaptatif pour diriger les messages codés vers diverses destinations.

TABLE 2 – Entête du message

Mono-diffusion	Indique si le message est en mono-diffusion ou non
Addr1	Tous les paquets dans le message codé sont originaires de Addr1
Addr2	ou Addr2 et ont comme destination Addr1 ou Addr2
Venant de	Le nœud voisin qui envoie le message
Nb. de paquets	Nombre de paquets dans le message codé
ID	L'identification du paquet
Source	L'origine ou la source du paquet
Existe à la dest.	Indicateur qui signale si le paquet existe à destination
Saut Suivant	Le nœud voisin sur le chemin vers la destination

L'entête des messages codés est montré dans le Tableau 2. La première partie de l'entête décrit le message codé, la corrélation entre les deux adresses Addr1 et Addr2, le nœud qui a transmis le message ainsi que le nombre de paquets codés dans le message. L'autre partie décrit les paquets codés et est répétée autant de fois qu'il y a de paquets dans le message codé. Chaque section représente des caractéristiques importantes du paquet ; son identité, son nœud source, l'indicateur "Existe à la dest.", qui indique si le paquet existe à la destination ou non ainsi que la destination intermédiaire prochaine du paquet. L'entête est mis à jour chaque fois que le message codé atteint un nœud intermédiaire et à chaque fois qu'il est codé avec un autre message. Le codage de différents types de messages ensemble et la mise à jour de l'indicateur sont détaillés dans ce chapitre.

Conclusion

Dans ce chapitre, nous avons introduit le CR à corrélation d'adresse comme un protocole qui améliore l'utilisation de la bande passante lorsque deux nœuds échangent

des informations. Nous avons démontré qu'avec l'utilisation d'indicateur adaptatif, nous sommes en mesure d'obtenir un gain en débit en effectuant des algorithmes simples pour garantir la décodabilité des messages codés aux nœuds récepteurs.

Chapitre 3
Décodage centralisé

Dans ce chapitre, nous détaillons l'approche de décodage centralisé où les nœuds intermédiaires décident de mettre en œuvre les fonctions de codage afin de réduire l'utilisation de la bande passante. Le décodage à lieu uniquement au niveau des nœuds de réception où les paquets émis et reçus sont utilisés pour le décodage. Ce processus nécessite l'utilisation de mémoires tampons et augmente de la complexité aux nœuds d'extrémités. Cependant, il n'y a pas beaucoup de charge sur les nœuds intermédiaires où seuls les algorithmes de codage sont mis en œuvre.

La contribution de ce chapitre est de plusieurs ordres ; premièrement, nous présenterons un modèle de décision au niveau des nœuds qui prend en compte les tailles des deux files d'attente. Deuxièmement, nous présenterons le graphique de codage, une nouvelle technique qui permet de compter le nombre de transmissions et les possibilités de codage lorsque le CRCA est utilisé sur des réseaux linéaires. Aux nœuds intermédiaires, nous étudierons les possibilités de codage et nous analyserons la taille des message codés. Troisièmement, nous étudierons les contraintes sur les tampons des nœuds d'extrémités ainsi que la complexité de décodage. Enfin, nous mettrons une restriction, appelée maturité, sur les paquets dans le réseau pour empêcher les paquets de vivre éternellement dans les messages codés.

Modèle de décision pour le décodage centralisée

Dans cette thèse, nous considérons un réseau linéaire composé de nœuds mettant en œuvre un processeur qui gère deux files d'attente ; la file d'attente de réception et la file d'attente de transmission. La file d'attente de réception est utilisée pour stocker tous les messages reçus. Au niveau de chaque nœud, le processeur examine les messages collectés dans la file d'attente de réception et étudie les possibilités de codage des paquets. La file d'attente de transmission contient les messages sortants prêts à être transmis au réseau.

Le processeur effectue principalement l'une des activités suivantes :

- La génération de paquets notée g
- L'activité de codage désignée par c
- L'envoi du message désignée par e
- L'activité de décodage notée par d

Au temps t, chaque nœud sélectionne une activité à exécuter à partir d'une liste d'activités assignée au nœud. Un modèle de décision est nécessaire à chaque nœud pour sélectionner la meilleure activité. Deux modèles de décision sont adoptés pour notre mise en œuvre, un modèle de décision déterministe pour les nœuds intermédiaires et un modèle probabiliste pour les nœuds d'extrémités.

L'activité de codage en profondeur

Dans cette section, on introduit la notion de *graphe de codage* qui sert à compter le nombre de transmissions nécessaires afin d'échanger un certain nombre de paquets. Le *graphe de codage* nous permet de démontrer le théorème suivant :

Théorème 1. *Étant donné un réseau linéaire $G = (N; E)$ avec deux nœuds d'extrémités et un nombre de paquets P échangés par les nœuds d'extrémités ($P/2$ paquets émis par chaque nœud d'extrémité) une limite supérieure sur la cardinalité des messages codés à une itération i donnée est donnée par :*

$$C(P,N,i) \leq 2 \times \min\left(i, \left\lceil \frac{N-2}{2} \right\rceil\right) + \max\left(\left(\min\left(i, \frac{P}{2}\right) - \left\lceil \frac{N-2}{2} \right\rceil\right), 0\right) \quad (3)$$

où N est le nombre de nœuds dans le réseau linéaire.

Temps de mise en mémoire tampon

Le temps de mise en mémoire tampon d'un paquet à un nœud est calculé comme étant la durée entre le moment où le paquet est reçu et mémorisé dans la mémoire tampon du nœud et le moment où ce paquet est utilisé pour la dernière fois dans un processus de décodage.

Nous avons montré par expérimentation puis justifié analytiquement que la taille de la mémoire tampon à chacun des nœuds d'extrémités doit être suffisamment grande pour mémoriser tous les paquets échangés depuis le début de la communication. Ceci vient du fait qu'il y a des paquets qui restent dans la mémoire tampon un temps égal au temps total de communication nécessaire pour livrer tous les paquets échangés. Cela se justifie par le fait que les premiers paquets générés pourraient rester dans des messages codés jusqu'à l'achèvement de la communication.

Le nouveau concept de vieillissement ou Aging

Définition 2. *L'âge d'un paquet est défini comme étant le nombre de fois où le paquet est impliqué dans une activité de codage, et l'âge d'un message codé est défini comme étant l'âge du paquet le plus vieux dans le message codé.*

Définition 3. *La maturité d'un réseau est défini comme étant l'âge le plus élevé autorisé pour tous les paquets dans le réseau.*

Pour chaque paquet i nouvellement généré, nous assignons la variable *age* initialement fixée à 0. Chaque fois qu'un message codé est créé et le paquet i est inclus dans ce message, le paramètre de vieillissement de ce paquet est incrémenté. Nous empêchons toute opération de codage qui implique un paquet ayant l'âge de la maturité.

Conclusion

Dans ce chapitre, nous avons montré l'importance du concept de vieillissement dans le codage réseau et le mécanisme utilisé afin d'empêcher les paquets de vivre indéfiniment dans le réseau. Avec ce concept, la complexité du décodage est considérablement réduite au nœuds d'extrémités ainsi que la quantité nécessaire de mémoire tampon. Nous croyons que le concept de vieillissement est une technique précieuse pour permettre le développement du CR en pratique et qui pourrait être mise en œuvre dans les futurs protocoles réseau.

Chapitre 4
Décodage distribué

Dans ce chapitre, nous introduisons un nouveau protocole de décodage distribué qui est mis en œuvre au niveau des nœuds intermédiaires du réseau. En particulier, nous proposons un algorithme de décodage des messages, qui améliore l'utilisation de la bande passante et réduit l'utilisation de ressources en permettant aux nœuds intermédiaires de participer à un sorte de décodage distribué. Le décodage distribué repose sur un mécanisme de temporisation des paquets dans les nœuds intermédiaires. Ces paquets sont utilisés pour réduire la cardinalité des messages codés transmis.

La contribution de ce chapitre est double ; premièrement, la création d'un mécanisme innovant de décodage où l'utilisation des ressources est réduite et distribuée entre les nœuds du réseau. Deuxièmement, l'étude de l'efficacité de l'algorithme de décodage distribué en terme de nombre total d'octets transmis dans le réseau, et la durée nécessaire de temporisation de chaque paquet dans les nœuds intermédiaires afin de garantir la décodabilité des paquets.

Notion de décodage distribué

Avec l'approche de décodage distribué, les nœuds intermédiaires sont également responsables de retirer la redondance inutile dans les messages codés sans compromettre le décodage au niveau des nœuds d'extrémités. Ce processus nécessite le sauvegarde des paquets à chaque noeud intermédiaire.

Avec le décodage distribué, chaque nœud intermédiaire maintient une mémoire tampon B utilisée pour stocker les paquets reçus. Lors de la réception d'un message, chaque nœud intermédiaire commence par la suppression des paquets qui sont déjà stockés au niveau du nœud, réduisant ainsi la cardinalité du message reçu. Si à la fin de ce processus la cardinalité du message est réduite à 1, le seul paquet restant est enregistré dans la mémoire tampon du nœud.

Cardinalité des messages codés

Pour vérifier l'efficacité du décodage distribué sur la cardinalité des messages transmis, des simulations ont été exécutées sur un réseau de 8 nœuds et la cardinalité des

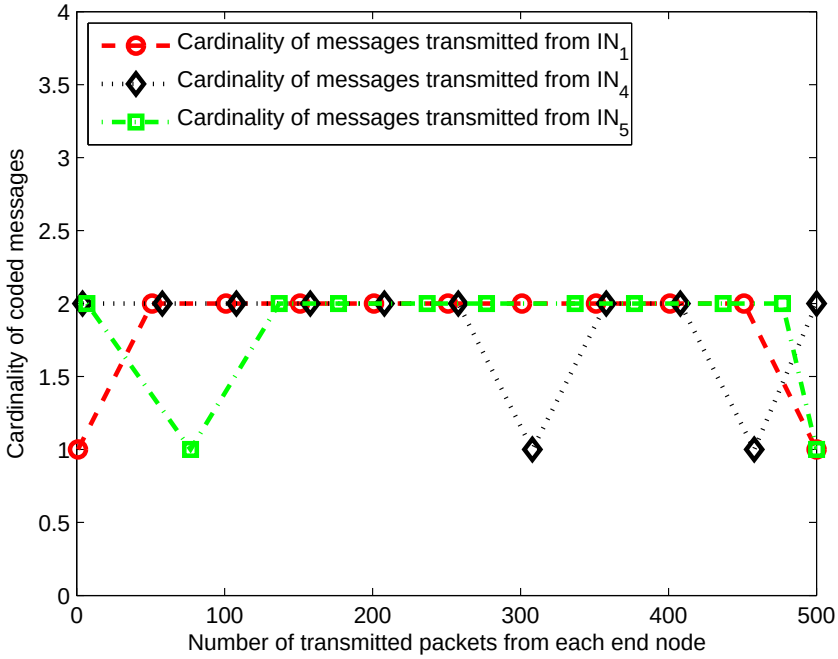


FIGURE 4 – Cardinalité des messages codés avec décodage distribué (réseau de 8 nœuds)

messages codés sur une sélection des nœuds intermédiaires est présentée dans la Figure 4. Nous observons que la cardinalité varie entre 1 et 2.

En effet, si nous travaillons avec le décodage centralisé ou distribué, un message codé contient au maximum un paquet qui est inconnu à sa destination. Dans le cas contraire, il sera impossible de décoder le message à la réception. Puisque dans nos réseaux linéaires, nous avons deux nœuds d'extrémités, nous pouvons avoir dans un message codé un maximum de 2 paquets qui sont inconnus au niveau des nœuds d'extrémités. Il est clair qu'avec le décodage distribué, la cardinalité des messages codés est minimale, et que les messages codés ne comprennent que les paquets qui doivent être livrés aux nœuds d'extrémités.

Durée de la temporisation

La durée de la temporisation représente un facteur important dans le CR et a des effets importants sur la qualité de service.

Le temps de mise en mémoire tampon d'un paquet à un nœud est calculé comme étant la durée entre le moment où le paquet est reçu et mémorisé dans la mémoire tampon du nœud et le moment où le paquet est utilisé pour la dernière fois afin de décoder un message.

Par simulation et d'une façon analytique, nous démontrons que le décodage distribué, comparé au décodage centralisé, comporte un gain de 93% sur la taille de la mémoire tampon. Ce gain est calculé comme étant le rapport entre la taille de la mémoire tampon nécessaire au niveau des nœuds d'extrémités et la taille de la mémoire tampon nécessaire avec le décodage distribué.

Conclusion

Dans ce chapitre, nous avons présenté une nouvelle solution pour décoder les messages dans le contexte du CR. Cette solution est basée sur une approche distribuée qui décode les paquets au niveau des nœuds intermédiaires. La procédure de décodage supprime, des messages codés, les paquets redondants et ainsi réduit le nombre d'octets échangés entre les nœuds d'extrémités. Des simulations ont montré une réduction de 64% du nombre d'octets transmis quand 1000 paquets sont échangés entre les nœuds d'extrémités dans un réseau linéaire de 8 nœuds.

Chapitre 5
Perte et reprise dans le codage de réseau

Dans ce chapitre, nous étudions l'impact de paquets perdus sur la taille de la mémoire tampon et sur la complexité au niveau des nœuds d'extrémités dans le cas du décodage centralisé et au niveau des nœuds intermédiaires dans le cas du décodage distribué. Nous proposons ensuite de nouveaux mécanismes pour permettre la récupération des paquets perdus. Comparé au protocole traditionnel de codage de réseau linéaire, nos mécanismes permettent d'obtenir une amélioration significative en termes de nombre de transmissions nécessaires pour récupérer la perte de paquets.

Nos contributions dans ce chapitre se résument comme suit :

- L'impact de la perte de paquets sur la capacité de décodage au niveau du récepteur et la perturbation causée par chaque perte sont étudiés et analysés.
- Deux nouveaux mécanismes de recouvrement, engagés par les nœuds du réseau sont proposés pour gérer la perte de paquets avec le CR : i) La demande de retransmission immédiate (IRR) qui utilise les informations extraites des messages non décodés pour demander la retransmission des paquets perdus, ii) le mécanisme de récupération de perte de paquets (BCSD) avec un nombre presque optimal de retransmissions.

Perte avec décodage centralisé

Les algorithmes présentés dans le Chapitre 3 sont utilisés pour compléter des simulations sur un réseau linéaire de 6 nœuds. Les statistiques concernant l'impact de la perte sur le décodage et sur le temps de mise en mémoire tampon au niveau des nœuds d'extrémités sont enregistrées et analysées. Notons que la perte de paquets est modélisée par une distribution Bernoulli. L'objectif de ces simulations est double. D'abord, comprendre la corrélation entre la perte de paquets et le nombre de paquets non décodés à la réception. Deuxièmement, identifier un processus de récupération des paquets perdus afin de minimiser le nombre de retransmissions.

Récupération de la perte de paquets : Pour remédier à la perte de paquets, deux nouveaux mécanismes sont détaillés dans cette section. Les performances de ces mécanismes sont évaluées en les comparant à une version révisée du protocole du CR linéaire. Ces mécanismes sont :

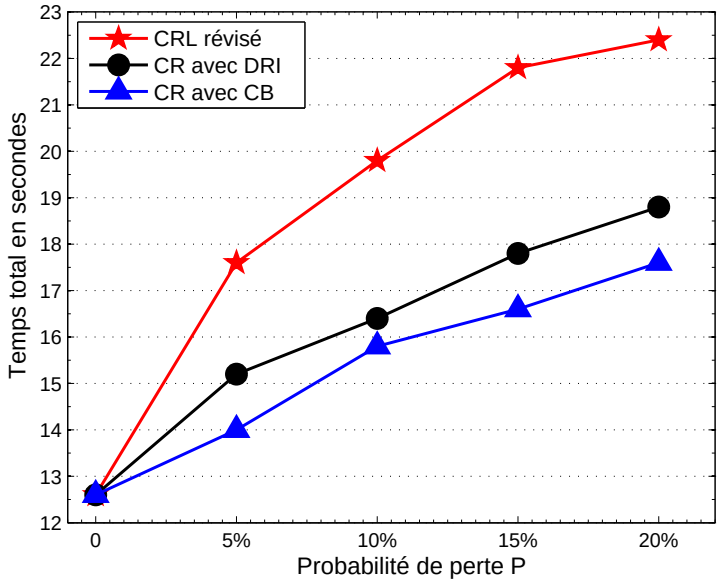


Figure 5 – Temps total de transmission nécessaire pour livrer 10000 paquets échangés

- *Demande de retransmission immédiate (IRR) :* Le nœud d'extrémité recevant un message sans toutefois être capable de décoder, arrive à identifier, par la lecture de l'entête du message, les paquets qui sont marqués comme "Existe à la dest." mais qui ne sont pas parmi les paquets mémorisés dans la mémoire tampon du nœud. Ces paquets sont considérés comme étant des paquets perdus, et une demande de retransmission est initiée par le protocole CR afin de récupérer ces paquets.
- *Couverture de base (CB) :* Le processus de récupération commence lorsque le nœud d'extrémité reçoit un message qui ne peut pas être décodé. A ce moment, le nœud s'attend à recevoir des messages indécodables supplémentaires. L'algorithme suit un mécanisme glouton qui tente, en un temps raisonnable, d'identifier un ensemble de couverture de base afin de minimiser le nombre de paquets à retransmettre.

La Figure 5 montre les résultats de nos simulations justifiant que les mécanismes proposés améliorent la qualité de service en réduisant le temps nécessaire pour livrer tous les messages.

Perte avec décodage distribué

Dans cette section, un nouveau mécanisme de codage est prévu pour détecter automatiquement et récupérer les paquets perdus dans les réseaux sans fil à pertes qui emploient le décodage distribué. Contrairement à la majorité des algorithmes de demande des répétitions automatiques (DRA) qui sont basés sur l'utilisation et la gestion des accusés de réception. Le mécanisme proposé est basé sur l'analyse des messages codés reçus et réagit automatiquement lorsque la perte de paquets est détectée. Ce mécanisme tire profit du fait que les nœuds intermédiaires gardent en mémoire tampon, pour une petite durée, les paquets transmis. Ces paquets mémorisés sont retransmis quand une perte est détecté.

Conclusion

Les modèles présentés dans les Chapitres 3 et 4 ont été utilisés dans ce chapitre afin d'étudier les pertes dans les réseaux et proposer des solutions de recouvrements tirant parti du CR. En effet nous avons montré qu'il y a assez d'informations dans les messages codés pour identifier les paquets perdus et proposer des mécanismes permettant aux nœuds du réseau de réagir face à ces pertes.

Conclusion

Dans cette thèse, nous avons proposé de nouveaux protocoles de CR basés sur la corrélation d'adresse appliquée sur les paquets échangés entre les nœuds d'extrémités. Ces protocoles sont basés sur deux méthodes de décodage, centralisée où le décodage a lieu au niveau des nœuds d'extrémités et distribuée, où tous les nœuds intermédiaires participent au processus de décodage. Nous avons commencé cette thèse par un aperçu général sur la théorie du CR, puis nous avons introduit dans le chapitre 2, le protocole CRCA qui garantit un certain niveau de qualité de service. Dans le chapitre 3, nous avons détaillé et étudié le protocole centralisé. Avec cette approche, les nœuds intermédiaires se concentrent sur le codage et la transmission des messages codés vers les destinations.

Ce protocole est étudié à partir de différentes perspectives et la notion de maturité est introduite afin de limiter la durée de vie des paquets dans les messages codés. Par la suite, nous avons développé dans le chapitre 4 le protocole distribué qui réduit la taille de la mémoire tampon nécessaire au niveau des nœuds d'extrémités en distribuant les ressources aux nœuds intermédiaires du réseau. Ainsi, avec le protocole distribué, une mémoire tampon est nécessaire au niveau des nœuds intermédiaires, mais dans l'ensemble, la taille de la mémoire tampon est considérablement réduite. L'idée de base derrière le protocole distribué est de forcer les nœuds intermédiaires du réseau à s'engager dans le processus de décodage en réduisant la cardinalité des messages transmis. Nous avons étudié dans le chapitre 5, l'impact de la perte et la récupération des paquets perdus pour les deux protocoles centralisé et distribué. Nous avons proposé des solutions permettant la récupération rapide et fiable de tous les paquets échangés dans les réseaux à pertes.

Introduction

In the current and next generation wireless network strategies, one of the major challenges in network communications is to accommodate the request for multimedia and data exchange service growth without compromising the various quality of service (QoS) requirements and without the need of network infrastructure changes. Moreover, communication systems are flooded by wireless networks and the nature of wireless communications requires an optimized use of the available bandwidth. The demand on bandwidth is thus increasing and traditional store and forward mechanisms used to transmit packets between end nodes become more and more bandwidth consuming compared to new techniques like network coding (NC). Thereby, NC where packets are combined together seems to be a promising idea to reduce the bandwidth usage while answering all communication requirements.

Many existing solutions for multiple access have been adopted to improve bandwidth usage. Such solutions are based on static medium access techniques like frequency division multiple access (FDMA), time division multiple access (TDMA) and code division multiple access (CDMA). The latter technique allows multiple senders to share the medium and transmit at the same time by partitioning the network capacity between the users. Bandwidth sharing was proposed as a solution to the wireless spectrum scarcity [1], and analog NC [2] allows multiple transmissions despite interference.

NC is a new networking technique, where multiple packets are combined together at some nodes in order to reduce the channel use in the network. Encoding and decoding require the use of algebraic algorithms where encoding accumulates various packets in fewer ones and decoding restores these packets. NC requires fewer transmissions to transmit all the data but more processing at intermediate and receiving nodes. In the

OSI layering model, each layer receives and transmits data to the adjacent layers; thus NC can be applied at any of the OSI layers. Physical layer network coding (PNC) is introduced in [8] to enhance the throughput performance of multi-hop wireless networks. The idea is to turn the broadcast property of the wireless medium into a capacity boosting advantage where simultaneously arriving electromagnetic (EM) waves are combined using GF(2) summation of bits at the physical layer. Media access control layer or simple MAC layer network coding presented in [9] works at the relay level by first encoding each data packet into a channel code and then taking bitwise XOR with the two channel codewords. At the network layer level, NC is introduced in [4] by linearly combining packets for retransmission. This technique achieves bandwidth usage minimization which opened the door for a wide research area and more than 3000 citations of the paper. In this thesis however, we focus on the network layer.

Given the potential of NC ability to work on different layers of the OSI model, cross-layer protocols appear to be a solution to improve the overall networking system performance. Although the traditional scheme of isolated layers in this case is broken, cross-layer models promote an important exchange of information between the layers for the sake of optimal solutions. The objective of this thesis is to introduce novelties to the NC paradigm with the intent of building easy to implement NC protocols. These protocols are designed to improve bandwidth usage, enhance QoS and reduce the impact of losing packets in lossy networks without adding overhead computation on intermediate or end nodes. From a methodological point of view, several challenges are raised in this thesis concerning details in the coding and decoding processes and all the related mechanisms used to deliver packets between end nodes. Notably, questions like the life cycle of packets in coding environment, cardinality of coded messages, number of byte overhead transmissions and buffering time duration are inspected, analytically derived and then verified through simulations. Moreover, by closely investigating the behavior of the lossy networks, we are able to propose novel mechanisms to overcome the loss and help reducing the overhead caused by the packet loss on the upper layer of the OSI model.

This thesis was carried out at the Institute of Electronics and Telecommunications of Rennes (IETR) and at the Lebanese International University (LIU). The thesis work has been structured to propose NC protocols suitable for both wired and wireless networks and respecting as closely as possible network requirements from QoS, with the

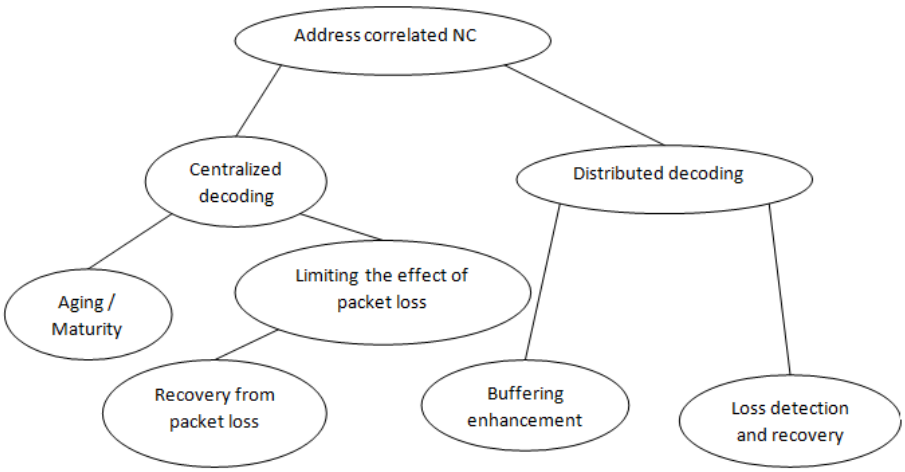


Figure 6: Thesis contributions

sole purpose of reducing bandwidth usage. Thus, the contributions of this thesis are summarized in Figure 6.

This manuscrit is organized as follows; in Chapter 1, the NC principles are introduced. In the first part, an overview of NC is conducted since triggered by the seminal work of Alshwede *et al.* [3] until the most recent publications on the subject. NC techniques are then detailed with the focus on linear NC and binary NC (based on XOR operations). These techniques are elaborated and supported by examples extracted from different scenarios to further help understanding the advantages and disadvantages of each of these techniques. In the second part, we elaborate on the benefits of NC from different points of view focusing on bandwidth usage and QoS. The chapter is concluded with a recap of its contents and a highlight on our main contributions to the subject.

In Chapter 2, a new address correlated NC (ACNC) protocol is presented together with the network model used for the implementation of this new protocol. In this model, intermediate nodes (relays) use a deterministic algorithm for activity selection and end nodes (senders and receivers) use a probabilistic algorithm for the same purpose. Coding and decoding activities are then detailed where the headers of both packets and coded packets are altered to support the proposed protocol. Two approaches using ACNC protocols are then introduced, the centralized approach where decoding is conducted at end nodes only and the distributed decoding approach where each node in the network participates in the decoding process. These approaches are studied from

buffering requirements and complexity points of view. The protocol is then extended to relay correlated NC where coding opportunity is enhanced especially between gateways of communicating networks. The chapter is concluded with examples illustrating the discussed protocols and their variations.

In Chapter 3, the centralized decoding approach is elaborated by first presenting the decision models and the procedure of decoding at end nodes. Moreover, the cardinality of received coded messages and the buffering requirements at end nodes are investigated. These investigations have led to the introduction of aging and maturity concepts. The importance of limiting the coding age of each packet in the network is supported by coding graphs and some theorems that give accurate computation of the number of transmissions required in exchanging packets between end nodes when using aging concept. Simulation results reveal some tradeoff between maturity and cardinality from one side and buffering and complexity from the other side.

In Chapter 4, distributed decoding approach is presented as a solution to reduce the overhead at end nodes by distributing the decoding process and buffering requirements to intermediate nodes. This approach is deeply studied from the following point of views; the cardinality of coded messages, the buffering time counting and the byte overhead transmissions. The chapter concludes with some simulation results and closed with a comparison between centralized and distributed approaches.

In Chapter 5, loss and recovery in NC are examined for both centralized and distributed approaches. For the centralized decoding approach, simulation of packet loss and decoding opportunity at end nodes is performed. Simulation results are then used to present two mechanisms to limit the loss impact. The first mechanism called immediate retransmission request (IRR) that directly requests lost packets from the sender when packet loss is detected. The second mechanism waits longer after a packet loss and works on optimizing the number of requested packets. The concept of closures and covering sets are introduced and the covering set discovery is conducted on undecodable messages to find the optimized set of packets to request from the sender in order to decode all received packets. For the distributed decoding, simulations on packet loss have led to the possibility of improving reliability by adding a hop-to-hop reliability mechanism that takes advantage of the NC itself and depicts loss without the need of an acknowledgment (ACK) mechanism. The chapter is concluded by simulations on lossy networks and discussions on the results where the proposed new mechanisms are

compared to traditional store and forward and to a revised version of linear NC.

Finally, a summary of the thesis is drawn in the general conclusion. Some perspectives and suggestions for future research works are also listed and discussed.

Publications

Journal papers

[A1]-Samih Abdul-Nabi, Ayman Khalil, Philippe Mary and Jean-François Hélard, *Efficient Network Coding Solutions for Limiting the Effect of Packet Loss.* Submitted to EURASIP Journal on Wireless Communications and Networking, 2015.

[A2]-Samih Abdul-Nabi, Ayman Khalil, Philippe Mary and Jean-François Hélard, *Aging in Network Coding.* Wireless Communications Letters, IEEE , vol.4, no.1, pp.78,81, Feb. 2015

International Conferences

[C1]-Samih Abdul-Nabi, Philippe Mary, Jean-François Hélard and Ayman Khalil, *Fault-tolerant minimal retransmission mechanism with network coding.* Accepted in the 23rd International Conference on Software, Telecommunications and Computer Networks (SoftCOM 2015). Split-Bol, Croatia. September 16, 2015.

[C2]-Samih Abdul-Nabi, Philippe Mary, Ayman Khalil and Jean-François Hélard, *Novel Distributed Decoding Scheme for Efficient Resource Utilization in Network Coding.* IEEE Wireless Communications and Networking Conference (WCNC). New Orleans, United States. March 9th, 2015.

[C3]-Samih Abdul-Nabi, Ayman Khalil and Jean-François Hélard, *Efficient network coding packet selection model for QoS-based applications.* IWCMC 2013 QoS-QoE Workshop (2013 QoS-QoE Workshop), Cagliari, Italy, Jul. 2013.

[C4]-Samih Abdul-Nabi, Ayman Khalil and Jean-François Hélard, *Routing coded messages in wireless networks.* 3rd International Conference on Communications and Information Technology (ICCIT-2013), Wireless Communications and Signal Processing (ICCIT-2013 WCSP), Beirut, Lebanon, Jun. 2013.

CHAPTER

1 Network coding principles

Instead of simply forwarding packets one by one, NC allows nodes to combine several packets into one coded packet to propagate through the network. NC basic idea is illustrated in the following examples. Figure 1.1 represents the famous butterfly network where node s multicasts packets to t_1 and t_2 through the nodes of the network. In this example, node 3 receives packets P_1 and P_2 to forward over the link 3-4. Traditionally, two channel uses are required to transmit the 2 packets. With NC, node 3 needs only to transmit a combination of P_1 and P_2 saving one channel use. By transmitting $P_1 \oplus P_2$ ($\oplus$ being the bitwise XOR operation), t_1 and t_2 are able to retrieve both packets, provided that t_1 and t_2 receive copies of P_1 and P_2 from other paths.

Another example showing the application of NC is illustrated in Figure 1.2 where 3 packets, P_1, P_2 and P_3 are transmitted from node T to all the other nodes of the network. The figure illustrates a network with one sender and three receivers with a random packet erasure process. Three cases are considered, a reliable transmission in Figure 1.2a, one different packet loss on each link in Figure 1.2b and Figure 1.2c. To recover the three packets P_1, P_2 and P_3, the scheme in Figure 1.2c using NC for the additional retransmission requires only one channel use instead of 3 channel uses with the scheme in Figure 1.2b, which does not use NC.

The mentioned examples shows that NC increases network capacity defined as follows:

Definition 1. *Network capacity is the amount of data that a network can carry.*

Even though the idea is simple, the implementation is far from being straightforward. Going back to the example of 1.1, NC can successfully reconstruct packets at t_1 and t_2 if these nodes receive enough information to decode the message coded by node 3. Otherwise, destination nodes will not be able to decode all packets and NC fails. However, in a wireless context, it is most likely that the destination node receives duplicate transmissions (due to the broadcast nature of wireless medium) by different nodes. This redundancy provides higher probability for successfully decoding messages. This

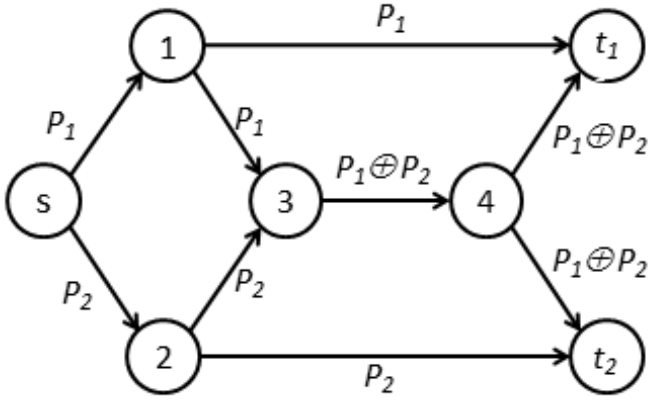
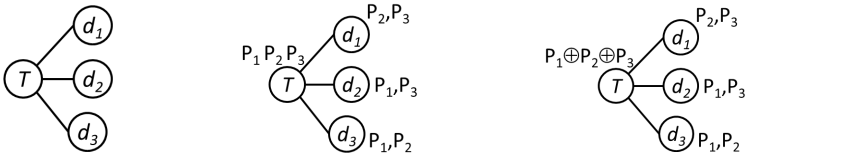


Figure 1.1: Butterfly example



(a) 3 channel uses (b) 6 channel uses (c) 4 channel uses

Figure 1.2: Network coding example

chapter is intended to provide the reader with an overview of NC history and the key concepts behind NC. The chapter is organized as follows; historical overview is presented in Section 1.1 where a path is drawn from the dawn of the first NC idea till the current NC research status. In the same section, examples are given showing the bandwidth saving introduced by NC. The coding and decoding techniques are detailed in Section 1.2. Section 1.3 summarizes the applicability of NC at different layers. Sections 1.4 and 1.5 show respectively the benefits and the usage of NC. The chapter concludes with Section 1.6.

1.1 Historical overview

Earlier before NC, packets were transmitted using the simple *store and forward* method. Relays, or intermediate nodes, wait for the reception of the entire frame, temporarily stored in memory, before being forwarded, if no errors are present, according to the relay routing table. However, when NC is involved, the forwarding concept has been

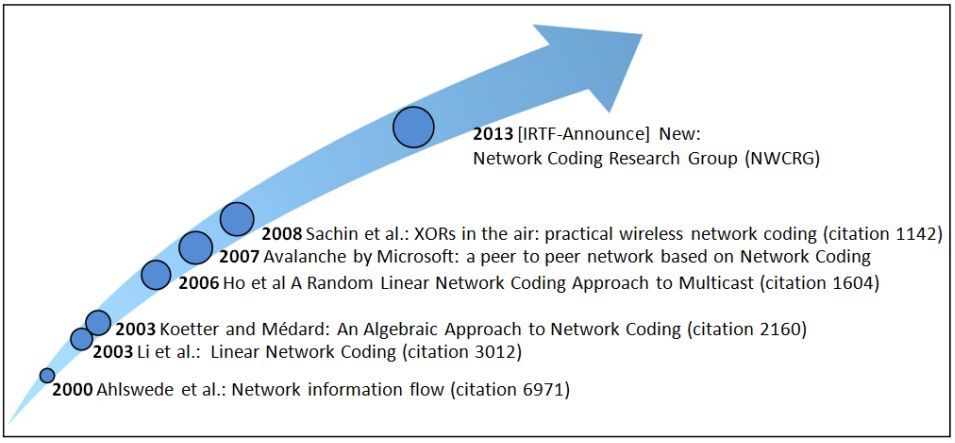


Figure 1.3: A glance on network coding

altered in such a way that some mathematical algebraic functions (addition, multiplication, combination...) are applied on the input packets right before being forwarded to the desired nodes. Nodes are allowed to modify packets as long as the receivers are able to retrieve the content of the original ones.

Figure 1.3 shows the progress of NC since it has been revealed as a concept in 2000 by Ahlswede et al. [3] till the creation of the NC research group (NWCRG) in 2013. In this figure, the number of citations indicated by Google Scholar is shown next to the key papers that marked the evolution of NC. As part of the Internet engineering task force (IRTF) responsible for several standards used by the Internet today (known as RFCs), NWCRG was created in 2013 with the aim to search NC principles and methods that can be beneficial for Internet communication. The first objective, as stated by the group, is to gather the research results and put forward the open questions related to NC in order to develop practical applications of NC that improve Internet communication. The other objective is to gather information on the existing practical implementations of NC, distill common functionalities and propose a path to standardization of NC-enabled communication. Worldwide, many large NC projects have been conducted by well-known institutes and financially supported by governments. We can list the following:

- Information Theory for Mobile Ad Hoc Networks (ITMANET). A joined project between STANFORD, MIT, California institute of technology and University of Illinois at Urbana-Champaign. ITMANET is a five-year project funded in 2006 and sponsored by the Defence Advanced Research Projects Agency (DARPA), an

agency of the U.S. Department of Defence.

- Network Coding for Robust Architectures in Volatile Environments (NCRAVE) is an European three-year project started in 2008 and partnered between different institutes including École Polytechnique Fédérale De Lausanne and Technicolor, a French private company known worldwide as a technology leader in the media and entertainment sector.
- Institute of NC (INC) established in 2010 and supported by the University Grants Committee of Hong Kong and the Chinese University of Hong Kong. INC is the largest engineering research project ever funded in Hong Kong. The mission of INC is to make Hong Kong a major long-term contributor in NC.

Ahlswede et al. [3] have shown that there is no information theory based reasons that relays only store and forward information, and, by allowing algebraic manipulations at relays, network throughput can be enhanced. Authors however have left the technical details of the algebraic manipulations as future work.

Koetter and Médard in [5] have given an important contribution in this direction, they have translated the NC problem to an algebraic problem by deriving transfer matrices and applying algebraic tools. In their work, they consider a node u as a node receiving a set of vectors, and willing to replicate these vectors at node v through a network where nodes between u and v perform linear NC techniques. They denote by $\underline{X}$ the matrix defined by the vectors received by u and by $\underline{Z}$ the set of vectors received by v and derived a transfer matrix M with $\underline{Z} = \underline{X}M$. Furthermore, they have made a connection between the network transfer matrix M which is an algebraic quantity, and the max-flow min-cut theorem which is a graph-theoretic tool. With their algebraic framework, they have shown the existence of a connection between u and v where the max-flow min-cut theorem is satisfied and the replication of $\underline{X}$ is assured by the non zero determinant of M.

The glory days for researchers in NC started in 2004 when the new emerged and novel approach to the operation and management of networks, specifically wireless networks with shared resources, has dominated as a hot topic. Many valuable algorithms have been published by several researchers. Jaggi et al. [10] have exposed new simple algorithms to be used in the design of linear multicast network codes operating over finite fields. Authors in [10] have proposed methods to select the coding vectors and others to validate the linear independence of these vectors. Meanwhile, and separately, Ho et al. [11]

have proved that each node could randomly choose coding vectors in a random and independent manner.

In the work of Fragouli et al. [12], a clear coding and decoding process, a list of NC benefits and applications for NC have been presented. Their work has clarified the linear coding approach and the selection of the coefficients required for the coding process. They have divided each coded packet into two parts: *i.e.* the encoding vector and encoding data. The encoding vector holds information required for decoding packets and the encoding data holds packets linearly encoded together. The decoding process requires solving a set of linear equations.

A series of publications by Pablo Rodriguez and Christos Gkantsidiś, two researchers at Microsoft laboratories, have led Microsoft to make a public announcement of a new technology. Avalanche was lunched as a peer to peer (P2P) network with improved scalability and bandwidth efficiency compared to existing P2P systems. Compared to BitTorrent, a content distribution protocol that enables efficient software distribution and p2p sharing of very large files, Avalanche improves some of BitTorrent shortfalls. Avalanche splits the file into small blocks for distribution. However, contrary to BitTorrent, peers transmit random linear combinations of these blocks along with the coefficient of the linear combination which reduces the complexity of managing blocks at peers. In [6], authors have proven through simulations that the expected file download time improves up to 2 to 3 times compared to sending non encoded information. Moreover, they have shown that NC improves the robustness of the system and is able to smoothly handle extreme situations where the server and nodes leave the system.

In 2008, Sachin et al. [7] published a complete NC based protocol named COPE. COPE inserts a coding layer, between the MAC layer and the Network layer of the OSI model, which is used to identify the coding opportunities available for saving transmissions. Thus, the services in layers above the coding layer are packet addressed (to each packet separately) while in the layers below the coding layer, a group of coded packets is being served. COPE combines 3 main properties:

- *Opportunistic Listening:* This technique takes advantage of the broadcast nature of wireless networks, by setting each node to overhear communications in the wireless medium and store overheard packets from neighboring nodes. Opportunistic listening requires omni-directional antennas to exploit the wireless broadcast property and additional storage for overheard packets.

- *Learning Neighbor State:* Each node sends periodically reception reports in order to inform its neighbors about the packets it had stored by annotating the packets it sends. If no packets are sent, the reception report is transmitted in a special packet format.
- *Opportunistic Coding:* In the coding process, each node uses its native packets stored in a local queue and overheard packets in an opportunistic queue to XOR as much packets as possible together in order to maximize throughput. Each node in the network benefits from the received reception reports by knowing the packets in their neighbors' queues, and uses this information to make the coding decision. An algorithm runs to ensure that neighboring nodes receiving the coded packets are able to decode. A simple example for this algorithm is illustrated when n packets are sent to n nodes, where a node can code n packets together if and only if all its neighbors have $n-1$ packets to ensure that they are able to decode the received coded packets and extract the needed one. A maximum benefit is observed when each of the n nodes is missing a different packet from the $n-1$ other nodes.

Simulations were run on a 20-node testbed running 802.11a. As a conclusion of their work, authors have determined the following facts:

- NC has practical benefits and can improve wireless throughput.
- COPE works well with congestion and improve throughput.
- In traffic with no congestion control (UDP), COPE's throughput improvement may substantially exceed the expected theoretical coding gain. Additional gain occurs because coding reduces the queue utilization thus reducing the probability of packet drop.

Although COPE was initially designed for stationary wireless mesh networks (WMN) where nodes are not resource-constrained, authors argue on the applicability of COPE beyond this limitation because of its flexibility to work with a variety of MAC protocols. They have also emphasized on the fact that modified versions of COPE can be applicable to wireless sensor networks (WSN) and cellular relays. It should be noted here that Ericsson has independently proposed a design for cellular relays with a subset of COPE's functionality, where the cellular relay XORs only duplex flows. However, one of the drawbacks of COPE is that nodes cannot rely completely on the reception reports because they might be lost in a severe congestion or delivered late due to network

traffic which affects the decoding probability. It should be noted that COPE provides algorithms to increase the decoding probability up to 80%.

The main question to rise when reviewing the literature of NC is where and when NC can be used. Advantages of NC are proven in theory however; this comes at the cost of additional constraints such that, additional computational complexity, storage use, etc. This tradeoff between throughput gain and resource requirements should be carefully studied when designing NC protocols. The limitations of the network and the required level of QoS are constraints among others that determine the validity of the coding and decoding process. In [13] authors have studied the encoding complexity in NC. They have restricted encoding to a subset of nodes (with higher computational power) while other nodes only forward packets without any encoding capability. In an acyclic [1] network where a source s needs to deliver h packets to k destinations, they have proven that the number of encoding nodes required to achieve the capacity of the network is bounded by h^3k^2 and it is in the same order when it comes to cyclic[1] networks. This leads to the conclusion that the total number of encoding nodes is independent of the size of the underlying network and depends only on h and k. The complexity of networks employing NC can be reduced by limiting the number of coding nodes. However, they have proven that, searching for the minimum number of encoding nodes required for a transmission is *NP-hard*.

In the following sections, we show examples of NC. We explain the max-flow min-cut theory for NC and we detail the coding and decoding processes. The benefits of NC are then listed along with areas where NC can be used.

1.1.1 Network flow

Before presenting the work of Ahlswede et al. [3], an introduction to some network concepts is needed.

Network concepts

A network is represented by a directed graph $G = (N, E)$ where N is the set of nodes and E the set of edges or arcs. An arc is a pair (i, j) where i is the origin node of the arc and j its destination node. Each arc is weighted with a capacity C_{ij}. In networking,

1. See Definition 6 in Section 1.1.1

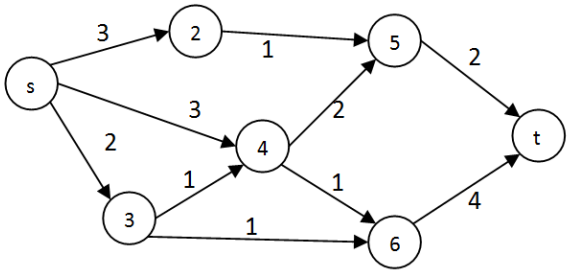


Figure 1.4: Example of a graph

arcs are links between points of communication and the capacity of a link (i, j) is the average number of bits that can be sent from node i to node j per unit of time. The set of nodes is divided into 3 subsets, the sources X, the sinks Y and the intermediates I. An example of such network is shown in Figure 1.4 where s and t denote respectively the source and the destination nodes.

To a network, we associate a flow $f \quad N \to \mathbb{R}^+$ where arcs are assigned a non negative value such that $0 \leq f_{ij} \leq C_{ij}$ for any arc (i, j) and $f^+(i) = f^-(i)$ for any node $i \in I$ with:

$$f^+(i) = \sum_{(i,j)\in E} f_{ij} \quad and \quad f^-(i) = \sum_{(j,i)\in E} f_{ji}$$

The condition $f^+(i) = f^-(i)$ is the flow conservation on intermediate nodes stating that the flow outgoing from an intermediate node (f^+) is equal to the flow incoming to the intermediate node (f^+).

Definition 2. *The value of a flow f, denoted by $val(f)$ is given by:*

$$val(f) = \sum_{x\in X} f^+(x) - f^-(x)$$

Definition 3. *A maximum flow or max-flow is a flow of maximum value.*

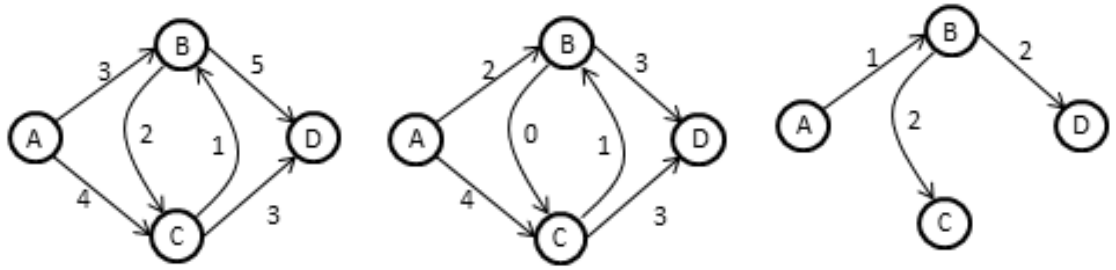
In the example of Figure 1.4 where $X = \{s\}$, the value of any flow f equals the flow on the arcs outgoing from node s. In networking, the main question is: what is the maximum value of a flow in a network?

Definition 4. *A path P_{uv} between two nodes u and v is a set of intersecting arcs connecting u to v without taking into consideration the orientation of the arcs. If the destination of every arc in set P_{uv} is connected to the origin of another arc in P_{uv}, then the path P_{uv} is said to be directed.*

Definition 5. *A cycle is defined as a directed path between a node and itself.*

Definition 6. *An acyclic network is a directed graph having no cycles. Otherwise, the network is said to be cyclic.*

Definition 7. *Let f be a flow in a network G. The corresponding residual network, denoted by R^f, is a network that has the same nodes as G, but the capacity RC^f_{ij}, named residual capacity, on every arc (i, j) in the residual network is given by $RC^f_{ij} = C_{ij} - f_{ij}$. Only arcs with non-zero capacity RC^f_{ij}, are included in R^f.*



(a) A network (b) A flow (c) The corresponding residual network

Figure 1.5: An example of residual network

Definition 8. *An augmenting path between two nodes u and v in the residual network is a directed path from node u to node v.*

As an example, Figure 1.5 shows a network (a), a flow equals to 6 (b) and the corresponding residual network (c). Labels on arcs of Figure 1.5a represent the capacity of arcs while the labels on the arcs of Figure 1.5b represent the flow on arcs. In this figure, a total of 6 units are sent from node A, the source, to node D, the destination. The flow conservation can be seen on node B where 3 units of flow are entering to node B and 3 units are leaving. With a capacity of 3 and a flow of 2, the residual arc (A, B) has a residual capacity of 1 as shown in Figure 1.5c. An augmenting path of value 1 is easily identified in the residual network shown in Figure 1.5c. An augmenting path of value n in the residual network means that we can push more flow along that path in the original network. In Figure 1.5c, an augmenting path of value 1 is identified between nodes A and D.

Augmenting by 1 the flow of the network as shown in Figure 1.6a gives the corresponding residual network shown in Figure 1.6b. Note that in this figure, no further augmenting paths can be detected and nodes A and D are no more connected in the residual network.

(a) Augmented flow (b) New residual network

Figure 1.6: Residual network of a maximum flow

Definition 9. *A cut $C = (S, \bar{S})$ is the set of arcs $\{(s, \bar{s})in\ E \mid s \in S, \bar{s} \in \bar{S}\}$ where $X \subseteq S \subseteq N - Y$ and $\bar{S} = N - S$.*

Definition 10. *The capacity of a cut C is defined by $Cap(C) = \sum_{(u,v)\in C} C_{uv}$*

Definition 11. *A minimum cut or min-cut is a cut of minimum capacity.*

As an example, consider the cut C shown in Figure 1.7. In this figure $S = \{s, 2, 3, 4\}$ and $\bar{S} = \{5, 6, t\}$. Arcs $(u, v) \in C$ are arcs (2, 5), (4, 5), (4, 6) and (3, 6) and the capacity of the cut is the summation of the capacities on the arcs in C which is 5.

Note that any flow in a network is subject to two constraints; the capacity constraint and the flow conservation constraints.

Lemma 1. *Given a network, for any flow f and any cut C $val(f) \leq cap(C)$*

Proof. Let $C = (S, \bar{S})$, since S is comprised of sources and intermediate nodes. Since intermediate nodes do not contribute to the flow:

$$val(f) = \sum_{x\in X} f^+(x) - \sum_{x\in X} f^-(x) = \sum_{x\in S} f^+(x) - \sum_{x\in S} f^-(x)$$

The flow on arcs with both endpoints in S contributes to both $\sum_{x\in S} f^+(x)$ and $\sum_{x\in S} f^-(x)$ thus do not affect $val(f)$. Therefore the only arcs with positive impact on $val(f)$ are the arcs originated in S and terminating in $\bar{S}$. We can say that:

$$val(f) \leq \sum_{(i,j)\in C} f(i,j) \leq \sum_{(i,j)\in C} C_{i,j} = cap(C)$$

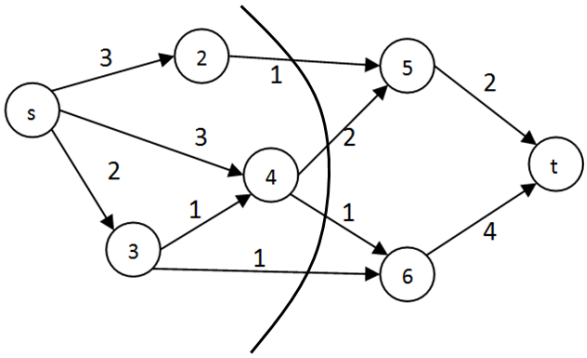


Figure 1.7: Example of a cut

■

Corollary 1 is a consequence of Lemma 1

Corollary 1. *Given a network, let f^* be the maximum flow and C^* the minimum cut. Then $val(f^*) \leq val(C^*)$*

Theorem 1. *known as the max-flow min-cut theorem [14]. The maximum value of a flow in a network is equal to the minimum capacity over all s-t cuts.*

Proof. Let f^* be the maximum flow in a network and let R^{f^*} be the corresponding residual network, then no augmenting path exists in R^{f^*} and the set of sources X is disconnected from the set of sinks Y. Let S be the set containing all nodes of X and the nodes reachable from any node in X and let $\bar{S} = E - S$. Therefore, $C = (S, \bar{S})$ is a cut and all arcs belonging to the cut C have residual capacity 0. That means in the original network the flow on each of these arcs equals to the capacity of the arc thus $val(f^*) = val(C)$. Or by Corollary 1 $val(f^*) \leq val(C^*)$ thus $val(f^*) = val(C^*)$ ■

In Figure 1.4, the minimum cut is obtained by setting $S = \{s, 2, 3, 4, 5\}$ and has a value of 4, thus the maximum flow that can be sent from s to t is 4.

The definition of a flow is not 100% accurate when it comes to communication networks. In such networks, we can no longer think of information as a real entity, since information needs to be transformed or replicated at the nodes. This is illustrated in Figure 1.1 where s is broadcasting information to both t_1 and t_2. Node 1 receives b_1 and broadcast copies on all outgoing links.

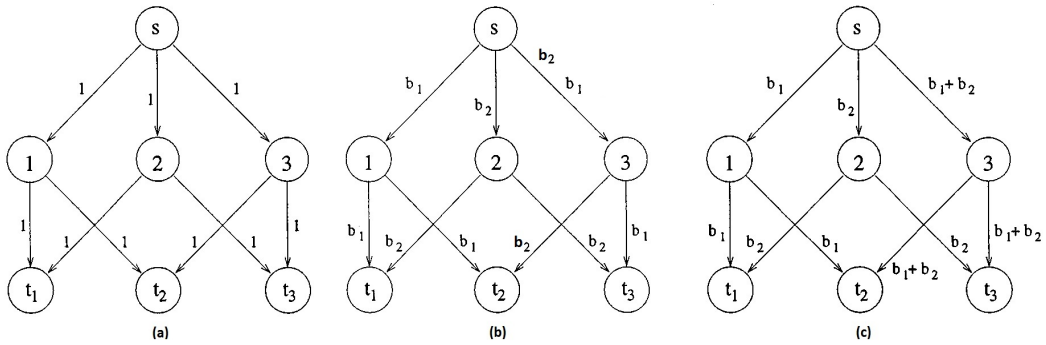


Figure 1.8: A one-source three-sink network

Network information flow

The journey of NC has started in 2000 with the work of Ahlswede et al. [3]. The authors leverage the function of a node from being a simple switch to a special case of encoder. Thus instead of replaying information from an input link to one or multiple output links, the node rather function as an encoder in the sense that it receives information from all the input links, encodes, and sends information to all the output links. As an illustration of their idea, consider the graphs of Figure 1.8 showing a network with one source s and three sinks t_1, t_2 and t_3. In this figure, the label on each link shows the capacity of the link considered here as the number of simultaneous transmissions or packets on the link.

The max-flow min-cut theorem is slightly modified in [3] through a conjecture defined as follows:

Conjecture 1. *Consider a graph with source s and sinks t_1, t_2, ..., t_l, the max-flow from s to all sinks equals to the minimum of the max-flow from s to every sink.*

In Figure 1.8, the max-flow from node s to the 3 nodes t_1, t_2 and t_3 is 2. Thus according to the max-flow min-cut theorem, two packets b_1 and b_2 can be sent to all destination nodes in one flow. However, sending b_1 and b_2 to the three destinations t_1, t_2 and t_3 requires 10 transmissions as shown in Figure 1.8(b) and cannot be performed in one flow (a second flow, shown in bold on the figure is needed to deliver both packets to all destination nodes).

Ahlswede et al. [3] have shown that coding within a network is needed to allow a source to multicast information at a rate of the capacity of the smallest cut between

the source and any destination. Figure 1.8(c) shows that by coding b_1 and b_2 using the XOR operation, only 9 transmissions are required to deliver both b_1 and b_2 in one flow leading to a saving of 10% in bandwidth. As a summary, for any random network where a source broadcasts information to destinations and independently of the network structure, the destinations get received information at the min-cut rate. This is possible when intermediate nodes operate in the same manner using NC.

1.1.2 Linear network coding

Instead of binary operations (*i.e.* Galois field (GF) of 2), linear network coding (LNC) permits moving to larger field sizes, where each data unit is processed using a finite field $\mathbb{F}_q$ with q a prime number or, considering a GF, $q = 2^m$ for some integer m.

LNC [4] released in 2003 by Li et al. complements the work of Ahlswede et al. In [4] authors have worked on networks with one source and multiple destinations and have proven that in the context of multicast, LNC allows to achieve optimality, which is the max flow from the source to each destination node. Authors have given a clear differentiation between commodity flow and information flow as follows:

- *Commodity flow:* flow is preserved on non-source non-destination nodes. The total volume of the outflow from a non-source node cannot exceed the total volume of the inflow.
- *Information flow:* The content of any information flowing out of a set of non-source nodes can be derived from the accumulated information that has flown into the set of nodes.

Constructively, they proved that by linear coding alone, the rate at which a message reaches each node can achieve the individual max-flow bound (*i.e.* between the set of sources and each destination). This result is considered stronger than the one stated in [3].

The authors have pointed in their conclusion the synchronization problem that needs to be addressed since intermediate nodes require some incoming data for possible coding which might be a serious problem for real-time applications. It should be noted here that the conference version of [4] was published in 1998, two years before the publication of [3] and that the work of Li et al. was cited by Ahlswede et al. while being submitted for publication.

Some authors have gone beyond coding over GF and proposed a complex field NC (CFNC) [15]. They have demonstrated that , CFNC always achieves throughput as high as 1/2 symbol per source per channel use. Authors have argued that their proposed CFNC approach is general enough to allow for transmissions from sources to a common destination as well as simultaneous information exchanges among sources.

1.1.3 Random network coding

Random Network Coding (RNC) was first purposed by Ho et al. [16] for multicast in lossless networks. RNC differs from LNC by the random selection of the encoding coefficients from some finite field $\mathbb{F}_q$. With RNC, the probability that all the receivers can decode the source packets (*i.e.* selected coefficients are linearly independent) is at least $\left(1 - \frac{d}{q}\right)^v$ for $q > d$, where d is the number of receiving nodes and v is the maximum number of arcs receiving packets with independent randomized coefficients in any set of arcs constituting a flow solution from all sources to any receiver. Similarly to LNC, a receiving node can decode received messages when the number of received linear independent combinations reaches the number of sources.

1.2 Coding and decoding techniques

In 2006, Fragouli et al. [5] have published *Network Coding: an Instant Primer* with the objective of showing the usage of NC in practice. With NC, each node is allowed to combine a number of received or created packets into one or several outgoing packets. Packets are required to be of equal length L and packets with different lengths are padded with trailing 0s. Each s consecutive bit is interpreted as a symbol over the field $\mathbb{F}_{2^s}$ with each packet consisting of a vector of L/s symbols.

1.2.1 Encoding

Original packets (*i.e.*, un-encoded packets) are denoted by $P^1, P^2, \ ..., P^n$, and each packet in the network is a linear combination of these original packets. To combine packets together, a sequence of coefficients $g_1, g_2, \ ..., g_n$ is generated and the combined packet, denoted by X, is a linear combination of the P^i packets using the g_i coefficient as follows:

$$X = \sum_{i=1}^{n} g_i P^i \tag{1.1}$$

Each sequence of coefficients $g_1, g_2, \ ..., g_n$ is used to create a combined packet. We denote by g^i the vector of an arbitrary sequence i of coefficients g_j $j = 1$ *to* n, and by X^i a combined packet created using g^i. The case where the field is $\mathbb{F}_2$, $g_i \in \{0, 1\}$ and the linear combination of P^1 and P^2 is $P^1 + P^2$ with the plus sign is the addition in $\mathbb{F}_2$ or bitwise XOR.

In what follows and later throughout this thesis, a packet refers to a non-coded piece of information and a coded message, or simply a message is a coded piece of information made of one or more packets coded together. Each coded message in the network holds two parts, the coefficients or the encoding vector, and the encoded data or the information vector. Receivers use the encoding vector to decode the information vector as it will be shown in Section 1.2.2. Moreover, for each packet P^i corresponds a unit vector $e_i = (0, \ ..., 1, 0, \ ..., 0)$ where the one is at the i^{th} position.

Encoding can be further performed on coded messages without the need of decoding first. However, the coefficients should always be computed with respect to the original packets $P^1, P^2, \ ..., P^n$. Assume a node receives coded messages $(g^1, X^1)..., (g^m, X^m)$, with g^i the encoding vector and X^i the information vector of the received coded packet created using g^i. The node needs to generate coefficients $h_1, h_2, \ ..., h_m$ and create a new coded message $(g^{'}, X^{'})$ as follows:

$$X^{'} = \sum_{i=1}^{m} h_i X^i \tag{1.2}$$

$$g^{'} = \sum_{i=1}^{m} h_i g^i \tag{1.3}$$

This operation can be repeated at any node in the network whenever the node receives a collection of new packets or coded messages.

1.2.2 Decoding

A destination or receiving node collects coded messages for possible decoding. Assume a node has received $(g^1, X^1)..., (g^m, X^m)$, to retrieve the original packets P^1, P^2, ...,

P^n, the node needs to solve the linear system of m equations and n variables where the unknowns are M^i.

$$X^j = \sum_{i=1}^{n} g_i^j M^i \quad j = 1 \; to \; m \tag{1.4}$$

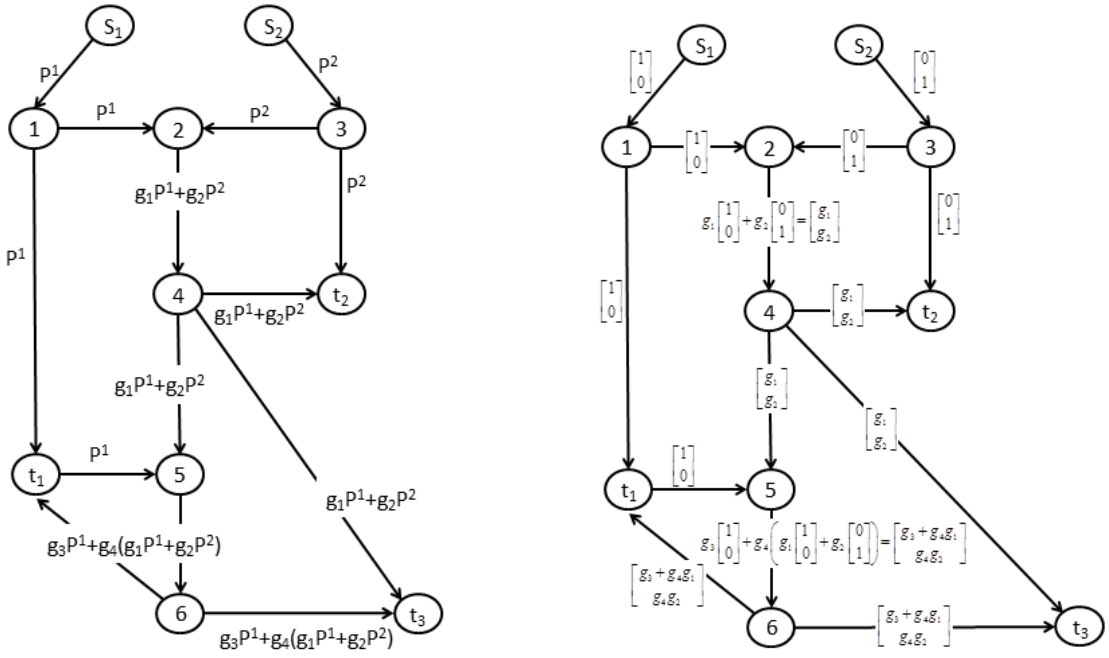
The node needs to have $m \geq n$ in order to decode the originally generated packets. The condition $m = n$ suffices if all the combinations are linearly independent. This is not an easy task when the network is decentralized. In [10], a distributed random linear NC approach is presented. Authors have shown that when the multiplying coefficients are selected at random, NC can achieve, over a sufficiently large finite field, the multicast capacity in an asymptotic way. It should be noted here that linearity makes coding and decoding easy to implement in practice. It is true that NC requires enhancing the computational capability of the network nodes but with the advancement of hardware technology, processing is less and less expensive. Thus NC is a win technology to answer the endless need of network usage and bandwidth utilization.

1.2.3 LNC example

To better understand the coding and decoding techniques with LNC, we consider a larger network than the butterfly introduced at the beginning of this chapter. In Figure 1.9, a two-dimensional NC is shown where sources S_1 and S_2 respectively broadcast packets P^1 and P^2 to three destinations t_i, $i = 1 \; to \; 3$. Considering a capacity of 1 on each link, the min-cut between the set of sources and each destination is 2 making it possible, using NC, to transmit two packets to all destinations in one flow. LNC occurs at nodes 2 and 5. Figure 1.9a shows the information vector while Figure 1.9b shows the coding vector. With LNC, addition and multiplication are performed over a finite field $\mathbb{F}_{2^m}$, where the resulting packets have the same length as the original packets. All three destinations receive enough information to decode received messages and retrieve packets P^1 and P^2.

1.3 Network coding at different layers

As we mentioned before, NC can be implemented in any of the communication network layers. Two popular layered models are the OSI and the TCP models [17] where



(a) Information vector (b) Coding vectors

Figure 1.9: Example of a network using LNC

the layers of these models are presented in table 1.1.

A suggestion to compare the implementations of NC on different OSI layers has been proposed in [18]. The authors suggest to compare the throughput of a wireless ad hoc network in a simulation using the optimized network engineering tools (OPNET). OPNET is used to create a simulated network where NC is implemented and tested on various layers of the OSI protocol stack. They have chosen the OSI model because it contains everything in the TCP model. The objective is, of course, to determine where NC can be done most efficiently. However, the authors in [18] did not perform such a comparison and to our knowledge, no such comparison is yet performed.

For researchers, the decision on which layer to apply NC depends on the required information needed for the coding and decoding methodologies. In the following, we analyze some of the works done on different layers.

Table 1.1: Popular layered models

TCP	**OSI**
Application	Application
	Presentation
	Session
Transport	Transport
Internet	Network
Link	Data link
	Physical

Application layer

Definition 12. *An overlay network [19] consists of a collection of nodes placed at strategic locations in an existing network. These nodes implement a network abstraction on top of the network provided by the underlying substrate network.*

Nodes of an overlay network have capabilities far beyond basic operations of storing and forwarding. In [20] NC has been conducted on the application layer of an overlay network in order to improve end-to-end throughput in a multicast topology. Authors in [20] have applied the mechanism of NC on intermediate overlay nodes. They claim that low layer NC (*i.e.* below the application layer) is only efficient on network graphs that conform to special patterns. Moreover, instead of modifying routers and switches to support NC, authors have proposed application-layer NC on overlay nodes as a solution to these problems where an overlay network offers flexibility in topology construction thus virtually building networks suitable for NC.

In [21], authors have proposed an application-layer cumulative coded Acknowledgments based on NC, that allows nodes to acknowledge network coded traffic to the sending nodes in a simple and efficient way. While NC applications try to avoid sending feedback, their proposed protocol encodes feedback hence exploits the benefits of NC.

Transport layer

In [22], authors have proposed a mechanism to incorporate NC into TCP, a transport layer protocol, with only minor changes to the protocol stack. In their approach, the sender transmits a random linear combinations of packets and a new interpretation of ACK is used to acknowledge coded messages but not necessary an original packet. Their solution introduces a new NC layer between the transport layer and the network layer of

the protocol stack. Authors have proven through simulation that the proposed solution lead to huge throughput gains over TCP in lossy links.

In [23] authors have proposed a new transport layer design for sensor networks where sensors are separated by unreliable, bandwidth limited communication and where packets may be dropped at any time. Authors in [23] have shown that the common practice of sending only the most recent observation is not optimal. Instead, they have derived necessary and sufficient conditions for an optimal linear combination of past and present observations. They have characterized the idea as *NC across time*, as compared to *NC across space* and that, in order to enhance performance.

Network layer

In [24], authors have used NC for content distribution in Vehicular Ad-Hoc Networks (VANET) and have proposed VANETCODE where static gateways, acting as servers, hold copies a file requested by moving vehicles. The file has been divided into blocks and RNC has been implemented at the network layer to code and transmit blocks to in range vehicles. Out of range vehicles continue to code and exchange blocks in order to reconstruct the requested file.

MAC layer

In [25], the NC model operates between the MAC and the network layer with the intention to code TCP acknowledgment and TCP data of the same flow together. This operation requires some interaction with the routing protocol so that both DATA and acknowledgment of the same flow use the same path between sender/receiver pairs. Authors have proven through simulation that their strategy offers a gain of 16% in terms of throughout and reduces by a factor of 10 the average time required to deliver acknowledgments with respect to MAC IEEE 802.11.

Physical layer

Physical layer NC has been introduced as a novel scheme in [26] with the intend to enhance the throughput performance of multihop wireless networks. As stated in [26], the basic idea of PNC is to take advantages of the natural addition, or the NC operation, that occurs when electromagnetic waves are superimposed on one to another. Note

that such operation requires synchronization between nodes so that packets arrive at the receiving relay with the packet boundary and symbol boundary aligned. Frame and symbol synchronization issues at transmitters for a common receiver can be found in [27]. In [28], authors have given a tutorial on the basic concept of PNC. They have also surveyed the recent key results with PNC and examined the synchronization challenges in PNC.

Cross layer NC

Cross layer NC or intra-session NC has been proposed to take the best of NC benefits. IN [29] authors have proposed a distributed way to implement novel congestion control for multicast flow where NC is used at the transport layer to adjust source rates and at the network layer to carry out network coded messages. In [30], authors have proposed a cross layer design across MAC and network layers to efficiently implement NC in order to extend the maximum throughput in wireless networks. Other cross layer NC applications can be found, among other, in [31], [32] and [33].

1.4 Benefits of network coding

NC initially targeted throughput gain as shown in [3–5]. It was used originally for wired then wireless networks and for multicast then even unicast [34]. Beside throughput, a list of benefits is presented:

- *Security:* it is difficult for an eavesdropper to understand what he can hear. Overhearing a coded message is always possible especially on wireless networks. However, with NC, decoding a message requires the processing of an additional piece of information that might never travel on the compromised link.
- *Robustness:* with NC, information in the network is duplicated and disseminated in the network while reducing the bandwidth usage. This spreading on information gives robustness to NC since any large enough collection of information might be used to decode transmitted information.

1.5 Applications using network coding

Applications where NC is used can be found in almost every aspect and type of networks. References listed in this section are examples of applications and not a complete list of the works done on the subject. For instance, [2] and [7] are examples where NC enhances throughput; [35]- [36] are examples where NC is employed to enhance reliability in a lossy networks. In these studies, block coding is used and the number of required transmissions of coded packets within a block is studied and compared to traditional automatic repeat request (ARQ) showing the benefits and capacity of NC to overcome lossy networks. NC is also used in distributed storage where it was shown that NC has the potential to provide dramatically higher storage efficiency for the same availability. With the advancement of cloud computation and social networks, distributed storage is thus becoming frequent with large scale deployment that absolutely needs to tolerate failure. In [37], NC is used in a framework of information exchange to repair failure in distributed storage system.

For data collection, NC seems to be a promising tool especially when collection is performed from harsh and extreme environments [38]. Partial network coding (PrNC) has been also introduced in [39] to solve the problem of removing obsolete information in coded data segments. Traditional NC requires solving of a linear problem in order to extract information from coded data which is almost unavailable at nodes of a distributed system. PrNC has been introduced as a novel technique to overcome this deficiency. Security has a major part of NC research and publications. In [40] authors have shown that random coding is secure with high probability if the coding is done over a large field. Models of NC is investigated in the case of wire-tapping [41] [42]. Byzantine and Pollution attacks is also examined using NC. Addressing security risks is an important topic in NC since with the process of mixing of information inside the network, a single malicious piece of information can end up contaminating all the information reaching a destination preventing decoding. Solutions to this type of attacks are given in [43] [44].

1.6 Conclusion

In this chapter, a general overview of NC was presented and the key research area in NC was identified. Moreover, the coding/decoding processes were well studied and

the interest in NC was identified. Among the field of interest, this thesis focuses on the application of NC to enhance throughput and to achieve reliability. In fact during the last three years, we have been concentrating on revealing all coding aspects in an address correlated NC protocol where only address correlated packets are subject to coding. This concept is thoroughly depicted in the next chapter.

CHAPTER 2

Address correlated network coding

2.1 Introduction

Address correlated network coding (ACNC) is a new NC protocol introduced in this chapter where intermediate nodes (or relays) mix packets together only if they are address correlated. Address correlation is defined as follow:

Definition 13. *Two packets are address correlated if the destination of any of the two packets is the source of the other one.*

In this study, linear networks are adopted for simulation and testing. This type of network can be seen as a static path within a large and diverted network as shown in Figure 2.1. In this figure, nodes 1, 2, 5, 8 and 10 form a linear network between a couple of devices exchanging information and connected respectively to nodes 1 and 10. Moreover, with static paths the opportunity for coding packets at intermediate nodes can be substantially improved. An application of linear networks is presented in [45] where the benefits of network coding over a wireless backbone were investigated from a theoretical perspective.

The contribution of this chapter is twofold; first, a novel forwarding approach to direct coded messages to various destinations is proposed. Second, an algorithm based on adaptive flag is presented to predict received packets at destination and improve

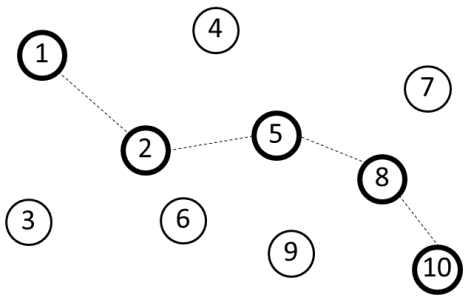


Figure 2.1: linear path in diverted networks

coding opportunities at intermediate nodes, in a way that guarantee the decoding of received messages at destinations.

This chapter is thus structured as follows; in Section 2.2, ACNC is detailed from a QoS point of view. The mathematical model to guarantee a certain level of QoS is thoroughly presented and a low cost NC algorithm is used at intermediate nodes. This algorithm selects candidate packets and coded messages to be further coded together. In Section 2.3, details of the coding process and the NC message header are presented and studied for variety of messages. Different coding and decoding techniques are also introduced in this section. Section 2.4 extends ACNC to relay correlated NC (RCNC) where some practical usages of such protocol are argued.

2.2 ACNC: Model and definitions

As mentioned before, the linear network shown in Figure 2.2 is formed when a set of end nodes, such as A and B, exchange information throughout a set of intermediate nodes. In such networks, users may exchange different types of information including video sequences, urgent messaging and files. This variety of message types requires prioritizing messages in order to maintain a high level of QoS. The emphasis here is on the selection of packets received and queued by a relay to be coded and released into the network.

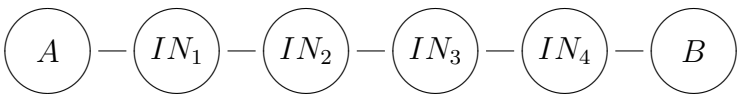


Figure 2.2: A network with four intermediate nodes

When a packet i arrives to an intermediate node IN_j, it is queued in the node and a timer t_i, initially set to 0, is associated to packet i. This timer tells how long the packet has been in the queue. The priority of each packet is indicated by Pr_i. Note that, Pr_i can be extracted from the packet header where priority is set by the end users, or, it can be set by the relay according to the characteristic of the packet. A binary variable y_i, initially set to 0, is also associated to each arriving packet; it is set to 1 when packet i is selected to be coded.

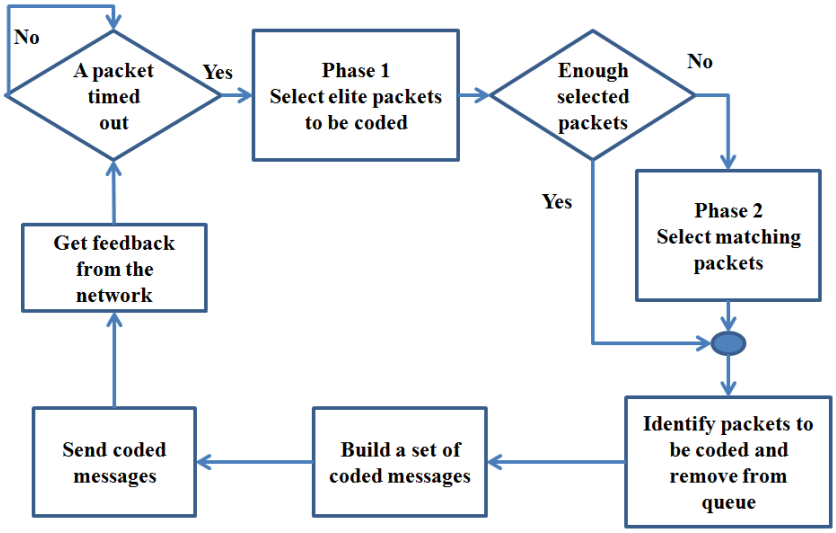


Figure 2.3: NC packet selection process diagram

The selection of packets to be coded and released into the network is completed in two phases and triggered when a packet times out in the queue. In phase 1, elite (high priority) packets are selected to be coded. If not enough packets are chosen, phase 2 is used to enhance the solution by adding to the selected set, candidates able to increase address correlation between the selected packets and thus improve the bandwidth usage. Once enough packets are selected in both phases, coded messages are built and sent into the network. The process then repeats itself with the time out value of a new packet in the queue. Figure 2.3 shows the packet selection process diagram.

NC requires a large pool of packets to select in order to increase coding opportunity and thus enhance throughput. Waiting for a packet to time out in the queue is necessary to accumulate packets in the queue without compromising the QoS. With limited queue size, when the queue occupation reaches a predefined threshold, phase 1 is triggered even if no packet is about to time out. This is needed to avoid packet queue overflow hence compromising the QoS.

Phases 1 and 2 are modeled as an optimization problem. The constraints and the objective function of such problem are presented in the next section. Afterwords, a low complexity algorithm inspired from the main features of the optimization scheme is presented and discussed. This latter algorithm is adapted for the selection and creation of coded messages in an acceptable amount of time.

2.2.1 Mathematical modeling of both phases

The optimization problem used in phase 1 to select elite packets is formulated as a minimization problem aiming to favor packets with high priority and those that are close to expire. Considering n packets in the queue, the minimization problem is given by:

$$\min \quad \sum_{i=1}^{n}(P_w(i) + E_w(i))y_i \tag{2.1}$$

$$\text{subject to} \quad C1 : \sum_{i=1}^{n} y_i \leq M_d \tag{2.2}$$

$$C2 : t_i - T(Pr_i) \leq y_i \times C \quad \forall\, i = \{1, ..., n\} \tag{2.3}$$

$$y_i \in \{0, 1\} \tag{2.4}$$

Some definitions are needed to clarify the constraints and the optimization function.

Definition 14. *The cardinality $|C_i|$ of a coded message C_i is defined as the number of packets coded within the message.*

The cardinality is equal to the number of non-zero elements in the coding vector, *i.e.* cardinality of 1 indicates none coded packet.

Definition 15. *The decoding capacity of a network is defined as the largest cardinality of coded message that can be decoded by the nodes of the network.*

Constraint C1: related to network capacity; Assuming that each relay in the network has limited computational power to solve a large system of linear equations (n equations of n variables), M_d is introduced as the decoding capacity of the network. With M_d given, the cardinality of the coded message cannot exceed the decoding capacity of the network. This leads to the first constraint; among the n packets in the queue, a maximum of M_d packets can be used to build the coded messages, with M_d updated dynamically by the relay. However, it should be noted that each packet in the coded message increases the size of the header by few bytes holding information about the incorporated packet, thus in practice, there is always a limit on the maximum number of packets to include in the coded message due a limitation in the size of messages supported by the medium.

Constraint C2: this constraint is related to priority management; given that each packet has a priority Pr_i, let $T(Pr_i)$ be the maximum delay allowed for packet i. A packet should be considered in a transmitted state at or before its maximum allowed time, that is, when the time t_i spent in the queue reaches the maximum delay $T(Pr_i)$.

The objective of constraint C2 is to force the selection of expired packets.

$$\begin{array}{lll} if \quad t_i \leq T(Pr_i) & \Rightarrow & t_i - T(Pr_i) \leq 0 \quad y_i \text{ can be 0 or 1} \\ if \quad t_i > T(Pr_i) & \Rightarrow & t_i - T(Pr_i) > 0 \quad y_i \text{ should be 1} \end{array}$$

The constant C should be chosen large enough to guarantee the validity of the constraint when $t_i - T(Pr_i)$ exceeds 1. Under this form, constraint $C2$ is fair enough to select packets that are required to be coded and removed from the relay. However, some factors should be taken into consideration to improve constraint $C2$:

1. Detecting packets that are about to be late, not only those that are late.
2. Taking into consideration the time needed for a coded message, that includes late or about to be late packets to travel to the next relay.

Therefore, $T(Pr_i)$ is defined as follow:

$$\begin{array}{rccl} T: & \mathbb{R} & \longrightarrow & \mathbb{R} \\ & Pr_i & \longmapsto & F(Pr_i) - Pr - t_d \end{array}$$

Where $F(Pr_i)$ is the maximum time the packet can spend in the queue, Pr is a precautionary constant and t_d the transmission delay to the next relay. t_d is determined dynamically depending on the network feedback and the round trip computation between the relay and its neighbors.

Objective function: the objective of phase 1 model, as indicated earlier, is to increase the amount of information in the coded message. The objective function takes into consideration the following properties:

- *Priority*: two packets i and j having priorities Pr_i and Pr_j respectively with $Pr_i < Pr_j$ but $t_i - T(Pr_i) = t_j - T(Pr_j)$ are treated equally by constraint C2 with no consideration to their priorities.
- *Expiry*: two packets i and j having priorities Pr_i and Pr_j respectively with $Pr_i = Pr_j$ and $t_i - T(Pr_i) < t_j - T(Pr_j)$ but both giving a negative left hand side of constraint C2 are treated equally by constraint C2.

Recall that Pr_i is used as the priority of packet i where priorities are between 1 and H with H indicating the highest priority. The priority weight for packet i is set as follow:

$$P_w(i) = -Pr_i \tag{2.5}$$

Given formulation (2.5), the lowest weight indicates highest priority. Similarly, the expiry weight of packet i is written as:

$$E_w(i) = T(Pr_i) - Pr_d - t_d - t_i \tag{2.6}$$

Formulation (2.6) indicates that the closer a packet to timeout is, the smaller the value of $E_w(i)$ is. Reaching a negative value, $E_w(i)$ indicates that packet i is delayed.

When phase 1 completes, packets are divided into two sets:

- $\overline{P_1}$ the set of packets already selected,
- P_2 the set of candidate packets.

The optimization problem used for phase 2 is formulated as a maximization problem aiming to find, among unselected packets the best matching with elite packets in order to maximize address correlation between selected packets thus reducing the number of transmitted coded messages. The formulation of the maximization problem is given in (2.7).

$$\max \quad \sum_{i=1}^{n} y_i \tag{2.7}$$

$$\text{subject to} \quad C1 : \sum_{i=1}^{n} y_i \leq M_d \tag{2.8}$$

$$C3 : \sum_{i=1}^{n} \min(1, |S_j - D_i| \times y_i y_j) < \sum_{i=1}^{n} y_i \ \forall j \tag{2.9}$$

$$C4 : \sum_{i=1}^{n} \min(1, |Cr_j - Cr_i| \times y_i y_j) < \sum_{i=1}^{n} y_i \ \forall j \tag{2.10}$$

$$y_i \in \{0, 1\} \tag{2.11}$$

The objective of phase 2 is to complement the set of selected packets $\overline{P_1}$ by selecting some packets from the set P_2 to maximize the cardinality of the produced coded messages. As done in phase 1, the solution should favor correlation between selected packets while maintaining the constraints on network capacity (see definition 1).

Constraint C3: this constraint is related to source/destination matching; selected packets should have correlated source and destination addresses so that they can be coded together and thus reducing the number of transmissions. If possible, for each selected packet, another packet should be selected with correlated addresses. We denote by S_i and D_i the source and destination of packet i and we say that two packets i and j have correlated addresses if $|S_j - D_i| \times y_i y_j = 0$. That is, either they are not selected by the system ($y_i y_j = 0$) or that the source of packet i matches with the destination of packet j ($S_j - D_i = 0$).

For a packet j, the left hand side of the constraint computes the number of non-matching packets with j. If the computed number is less than the number of packets, then the constraint guarantees the presence of at least one match for packet j.

Constraint C4: other criteria might be required to allow packets to be coded together. Constraint C3 is generalized to support any grouping criteria (other than address correlation) between selected packets. Denoting by Cr_i the criteria of packet i, constraint C4 guarantees, to a certain level, the selection of pairs of packets with the same criteria. The right hand side of the constraint can be replaced by ≤ 0 if all selected packets are required to have the same criterion.

2.2.2 Complexity analysis of the proposed model

The proposed model for phase 1 adopts a binary integer programming (BIP) model. In such a model, each decision is represented with a binary variable: setting the variable equal to 1 corresponds to making the "yes" decision, *i.e.* the corresponding packet is selected for coding, while setting it to 0 corresponds to the "no" decision, *i.e.* the corresponding packet is not selected for coding. Constraints are then formed to correspond to the effects of the decision. Computer codes for BIP algorithm are now commonly available in mathematical programming software packages. Traditionally these algorithms have been based on branch and bound B&B technique and its variation [46]. The field of BIP has witnessed remarkable improvements over recent years. Two of the

biggest contributors have been heuristics [47] and parallelism [48]. The main source of parallelism is the fact that different nodes in the BIP search tree can be processed independently. Given the tradeoff between speed and hardware cost, the parallelism technique requires powerful hardware and high energy consumption. On the other hand, the proposed model for phase 2 is a nonlinear optimization problem with linear objective function. These types of problems are generally classified as NP-Hard (non-deterministic polynomial-time hard). In real time applications, there is a need to replace traditional methods by iterative algorithms able to improve the performance and runs within a reasonable amount of time to find a solution.

2.2.3 Low cost algorithm

In this section, a low cost iterative algorithm is presented. The algorithm has the same objectives as the models proposed in Section 2.2.1.

- Improve throughput with NC.
- Improve QoS with the selection of packets to be coded.

Our low cost algorithm using network coding (LCANC) is listed in Algorithm 1. LCANC targets packets that are about to expire to match them together. If no match is found, the algorithm uses the remaining available time to match the selected packets with non-expired packets if available. No transmission of non-coded packets occurs unless a packet expire and cannot be matched with any other packet in the queue.

The LCANC is compared with a traditional version of NC (TNC) listed in Algorithm 2. TNC does not take into consideration QoS and performs coding between the packet in the top of the queue and the first existing match in the queue. If no match is found the packet is transmitted individually.

2.2.4 Simulation results

To validate the efficiency of the proposed approach, a small network made of two linear networks, A, IN_1, IN_2, IN_5, C and B, IN_3, IN_2, IN_4, D is implemented as shown in Figure 2.4. In this network, 4 users are exchanging different types of data such as video conferences, voice and file download. The simulation is done to test the global queuing system of the relay IN_2. Packets arrive to the relay following a stochastic probability distribution and are stored in a buffer. Considering that this buffer has a

Algorithm 1 LCANC

```
1: repeat
2:     repeat
3:         Pick one packet from expired
4:         if matched with expired then
5:             Use NC
6:         else
7:             if matched with nonexpired then
8:                 Use NC
9:             else
10:                Send individual
11:            end if
12:        end if
13:    until no Expired or network capacity reached
14:    repeat
15:        Pick one packet from nonexpired
16:        if matched with nonexpired then
17:            Use NC
18:        end if
19:    until Expired or network capacity reached
20: until Network stop
```

Algorithm 2 TNC

```
1: repeat
2:     repeat
3:         Pick first packet from the queue
4:         if expired then
5:             Send individual
6:         else
7:             if another packet in the queue then
8:                 if Correlation between addresses then
9:                     Use NC
10:                else
11:                    Send individual
12:                end if
13:            end if
14:        end if
15:    until Network capacity reached
16: until Network stop
```

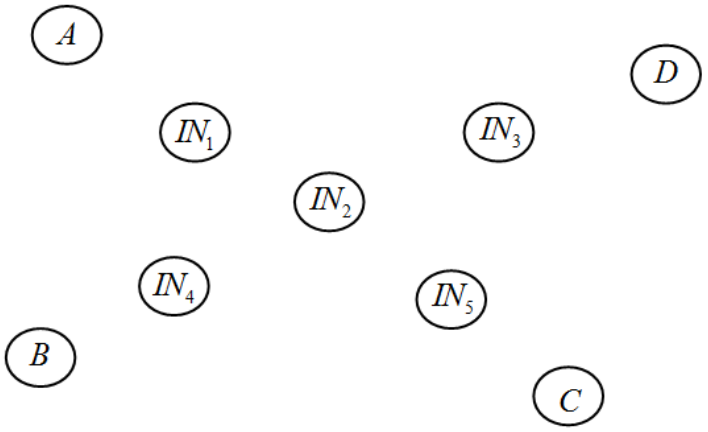


Figure 2.4: Two linear networks intersecting at relay IN_2

limited size, selecting packets from this buffer is done according to three techniques that will be listed shortly. Some of the selected packets are coded together; others are sent individually. Selected packets (coded and non-coded ones) are then sent to the output interface. The output interface has a limited capacity that is set equal to the capacity of the network. This capacity is made variable to simulate the network congestion. Three algorithms are used in our simulation to forward packets. The first algorithm considers the traditional store and forward (SF). The second algorithm is the TNC and the third algorithm is the proposed LCANC.

The performance of SF, TNC and LCANC in terms of the number of transmissions are plotted in Figure 2.5. The network has a capacity of 8 packets. It is clear that TNC has a higher rate of transmitted packets compared to traditional store and forward. As described earlier, TNC guarantees decoding at the end user since packets are coded as function of some packets available at the end users. The figure shows that, using TNC, the throughput is enhanced and the number of transmitted packets exceeds the network capacity. Recall that network capacity is defined as the amount of data that a network can carry, with NC, each coded message holds multiple packets making it possible to exceed the network capacity. When the plot reaches values above SF, TNC shows that coding is done and that throughput gain occurs. Being below SF means that the queue at the output is not at its capacity and that the switch has fewer packets to send than the capacity of the network allows. The major gain in throughput is observed for LCANC that remains always above SF and TNC indicating that the algorithm always uses its

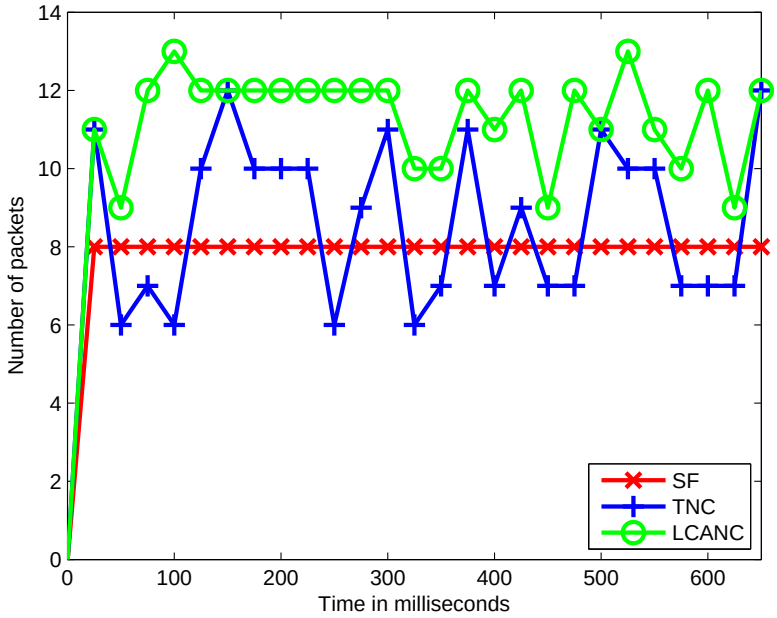


Figure 2.5: Instantaneous number of transmitted packets

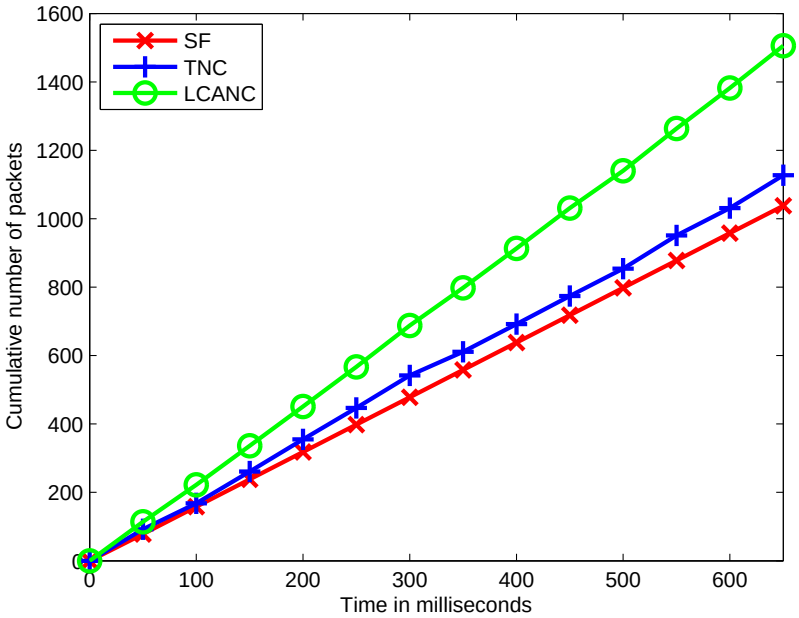


Figure 2.6: Cumulative number of packets transmitted

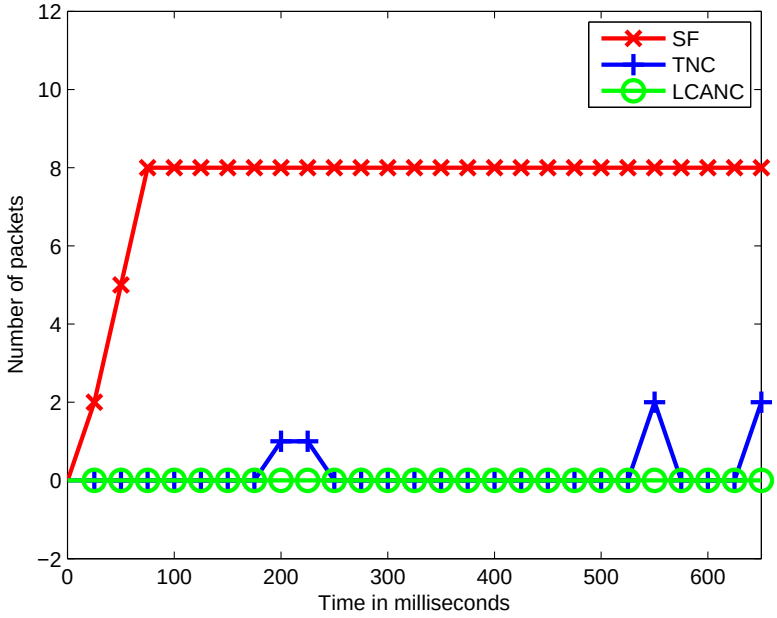


Figure 2.7: Instantaneous number of delayed packets in the queue

coding capacity to code as much packets as possible.

Figure 2.6 shows the cumulative number of packets sent with respect to the time in milliseconds. The cumulative number of packets at time t is defined as the total number of packets transmitted by all the nodes of the network since the beginning of the simulation until time t. It is clear that for a specific time, the number of packets sent by the proposed LCANC algorithm is larger than the number of packets sent by any other method. However, this comes at the cost of increasing the processing capability of the switch which is a fair compromise with respect to the throughput improvement achieved.

Figure 2.7 shows the number of packets that are delayed with each algorithm. A packet is considered to be delayed if the time spent in the queue exceeds the maximum allowed. While SF plot shows that after a while, all packets in the network are delayed, the proposed method shows that no delayed packets are transmitted. This proves that the proposed method manages the selection of candidate packets to be coded in an efficient way.

2.3 A practical implementation of ACNC

ACNC adds a layer between the data link and the network layers of the OSI networking model. In general, ACNC generates a coded message made of packets coded together using XOR operation on bits of the packets. A header is added to each coded message describing each packet coded within the message and the correlation between these packets. ACNC also uses adaptive flag to steer coded messages to various destinations. Adaptive parameters are used in [49] where NC is employed only for packets belonging to the same generation. In their work, feedback from the receivers to the source is used to adaptively adjust the generation size and thus maximize the number of packets successfully decoded at the receivers. In this section, we detail the process of receiving, coding and forwarding messages using ACNC.

2.3.1 Context and network

We work in the context where a switch (or an intermediate node) receives packets and coded messages that need to be forwarded to their destinations. Our receiving, coding and forwarding process (RCFP) is used whenever the node receives or has in its queue, potential messages to code and transmit. RCFP in fact is divided into two parts; receiving process (RP) which is used to handle received messages and immediately updates the corresponding adaptive flag, and the coding and forwarding process (CFP) that uses the updated adaptive flag to code and transmit messages. RP/CFP protocol is placed between the Data Link and the network layer in the communication layering stack.

2.3.2 Receiving, coding and forwarding process

The process of coding and forwarding messages requires adding a header to the created coded messages. This header allows receiving nodes to identify packets included in the coded message and consequently route copies of the coded messages to various destinations. The receiving, coding and forwarding process is described in this section.

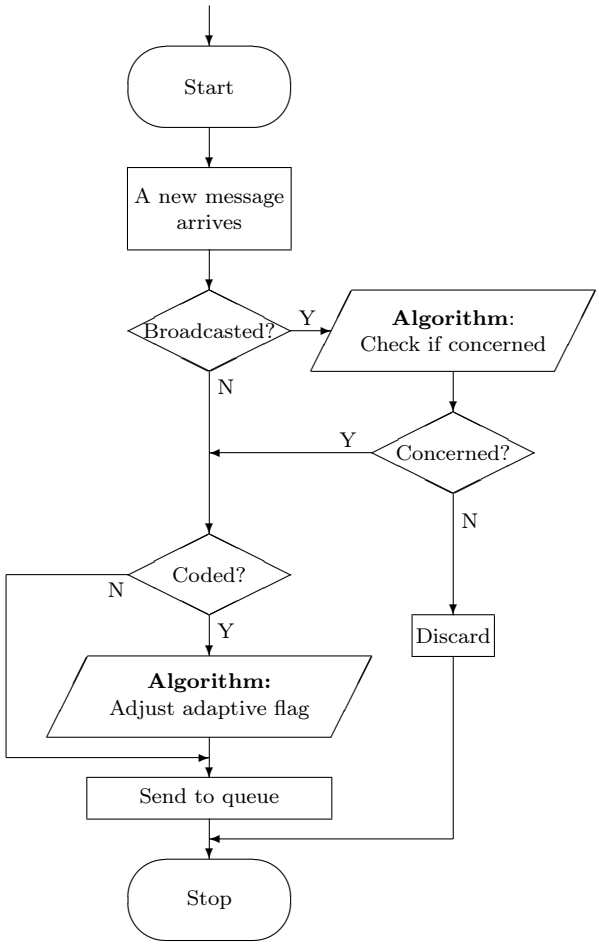


Figure 2.8: Receiving Process (RP)

Receiving process (RP)

A node can receive coded or uncoded messages. If a message is received by the node in a broadcast mode, the node needs to check whether it is concerned or not by that message. A node is concerned by a message if it is either the destination of the message (*i.e.* the destination of at least one of the coded packets in the message) or if it is a relay on the path from the source to the destination of one of the packets coded in the message. If the node is concerned by the received message, it immediately updates the adaptive flag of the received message and sends it for queuing. The receiving process is presented in Figure 2.8.

Coding and Forwarding Process (CFP)

An intermediate node might decide to forward a received message or to code messages together in order to improve throughput. The decision is based on QoS constraints as described in section 2.2. The first step in CFP is to validate the capability of end users to decode each message. Capability of decoding is validated using adaptive flag. Once the possibility of decoding is confirmed, CFP performs a coding process and creates a coded message. The coded message is prefixed by a header that describes each packet included in the coded message and holds the corresponding adaptive flag. The message is then transmitted to the network. The overall CFP process is presented in Figure 2.9.

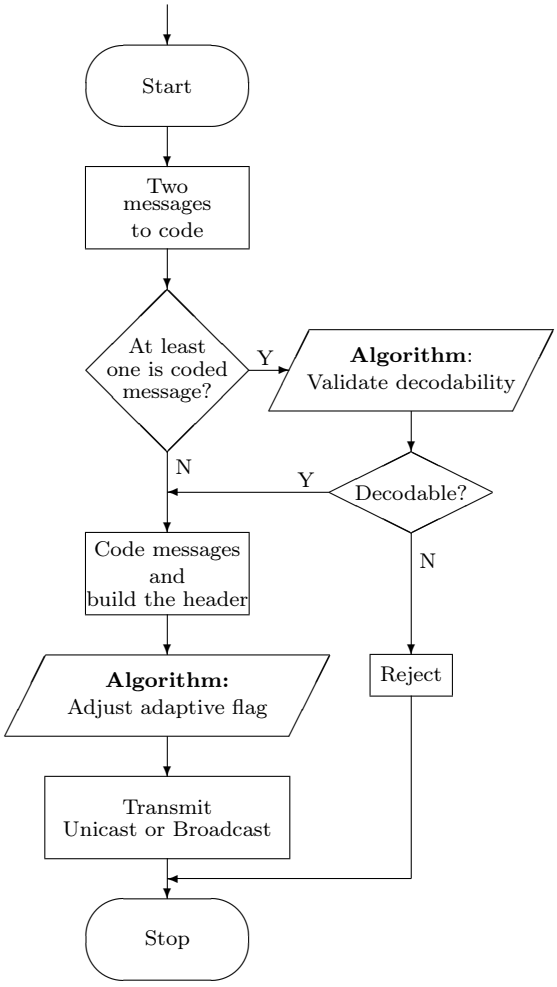


Figure 2.9: Coding and Forwarding Process (CFP)

Table 2.1: Message header

Unicast	Tells if the message is a unicast of a broadcast
Addr1	All packets in the coded message have Addr1 (respectively Addr2)
Addr2	as origin and Addr2 (respectively Addr1) as destination
Coming from	The neighboring node sending the message
Nb. of packets	Number of packets coded inside the message
ID	Identification of the packet
Source	Origin or source of the packet
Exist at dest.	Adaptive flag that tells if the packet exists at destination
Next hop	Neighboring node on the path to destination

2.3.3 Message header

The header of a message is shown in Table 2.1. The first part of the header describes the coded message, which indicates that it is exchanged between end nodes identified by Addr1 and Addr2 and released by node "Coming from". Note that Addr1 and Addr2 are the correlated addresses and can be interchanged . The other part describes the packets coded together and is repeated as many times as there are packets in the coded message. Each section lists four important characteristics of the packet; its ID, its source node, the adaptive flag "Exist at dest." which indicates whether the packet exists at destination or not and the next hop of the packet. The header is updated each time the coded message reaches an intermediate node and each time it is coded with another message. The coding of different types of messages together and the update of the adaptive flag is detailed in subsequent sections. We then describe the forwarding and receiving processes and the usage of the adaptive flag to guide the messages to its destinations.

2.3.4 Coding two packets

Switch IN_i decides to code two packets as described in Table 2.2. These packets are received from neighboring nodes as shown in Figure 2.10

Packets 123 and 124 are XORed together to form a new coded message that will be transmitted from switch. The header of the coded message is listed in Table 2.3a.

After building the coded message, fields such unicast, "Exist at destination" and next hop needs to be updated. Next hop is determined by looking at the routing table of the node. The next hop of packet 123 is IN_{i+1} and the next hop of packet 124 is

Table 2.2: Packets to be coded by node IN_i

Unicast	Yes
Addr1	A
Addr2	B
Coming from	IN_{i-1}
Nb. of packets	1
ID	123
Source	A
Exist at dest.	No
Next hop	- - - -

Unicast	Yes
Addr1	B
Addr2	A
Coming from	IN_{i+1}
Nb. of packets	1
ID	124
Source	B
Exist at dest.	No
Next hop	- - - -

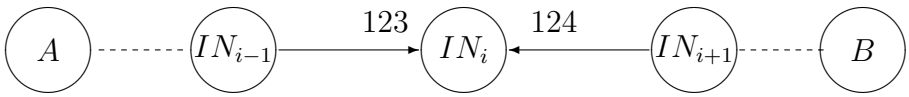


Figure 2.10: A linear network showing a coding opportunity

IN_{i-1}. To set the "Exist at dest" flag of each packet, the node compares the "coming from" value in the received message and the "Next hop" value in the generated message. A match between those values is an indication that a copy of the packet is already in its way to the destination and the "Exist at dest" flag is set to true, otherwise the flag is set to false. When all packets' specific headers are updated, "Unicast" is set to true if at most one packet is unknown to the destination. In this case, the message is unicasted to the destination of that packet. If more than one packet is unknown to its destination, the message should be routed to more than one destination and the "Unicast" field is set to false. The updated header is shown in Table 2.3b and the transmitted message is shown in Figure 2.11.

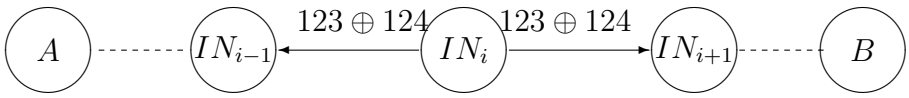


Figure 2.11: Transmitted coded message after coding two packets

Table 2.3: New created coded message

(a) New header of a coded message creating by coding two packets

Unicast	- - -
Addr1	A
Addr2	B
Coming from	IN_i
Nb. of packets	2
ID	123
Source	A
Exist at dest.	- - - -
Next hop	- - - -
ID	124
Source	B
Exist at dest.	- - - -
Next hop	- - - -

(b) Updated header of the coded message before transmission

Unicast	No
Addr1	A
Addr2	B
Coming from	IN_i
Nb. of packets	2
ID	123
Source	A
Exist at dest.	No
Next hop	IN_{i+1}
ID	124
Source	B
Exist at dest.	No
Next hop	IN_{i-1}

2.3.5 Receiving a message by unconcerned node

When node IN_j receives a broadcasted coded message it performs Algorithm 3 to test whether it is concerned or not by that message. The intermediate node is concerned by a message if it is the next hop of any of the packets contained in the message.

Algorithm 3 Test if concerned node

```
1: loop on all packets in the message
2:     if Next hop = IN_j then
3:         Set to Concerned
4:         Stop
5:     end if
6: end loop
7: Discard the message
```

2.3.6 Receiving coded message by concerned node

When node IN_{i+1} receives a broadcasted coded message and it is a concerned node (one of the next hops is indeed IN_{i+1}). It performs Algorithm 4 to update the header information of the received node. In line 4 of Algorithm 4, the "Exist at dest." is set to

true when a packet is coming from a node considered in the routing table as the next hop of the packet in its way to destination.

Algorithm 4 Update adaptive flag

```
1: loop on all packets in the message
2:     NH ⟵ next hop according to routing tables
3:     if NH = Coming From then
4:         Exist at Dest. ⟵ Yes
5:     else
6:         Set next hop
7:     end if
8: end loop
```

2.3.7 Update of the flag "Exist at Dest."

A coded message is created and broadcasted by node IN_{i+1} after coding packets 224 received from IN_i and 225 received from IN_{i+2}. Node IN_i decides to code a packet with a the coded message. When node IN_i receives the broadcasted coded message it runs Algorithm 4 to update its header information including adaptive flag. The context of the transmission is shown in Figure 2.12 and the headers are shown in Table 2.4.

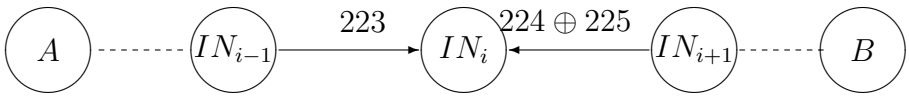


Figure 2.12: Coding one packet and one coded message

Coding feasibility

Before presenting the coding scheme, the coding node should validate the decoding feasibility of the resulting message at each destination. Referring to Table 2.4, the possibility of coding is justified as follows; end node A has messages 223, 225 and missing message 224 while end node B has messages 224, 225 and missing 223.

Table 2.4: A packet and a coded message to be coded by node IN_i

Unicast	yes
Addr1	A
Addr2	B
Coming from	IN_{i-1}
Nb. of packets	1
ID	223
Source	A
Exist at dest.	No
Next hop	IN_i

Unicast	Yes
Addr1	A
Addr2	B
Coming from	IN_{i+1}
Nb. of packets	2
ID	225
Source	A
Exist at dest.	Yes
Next hop	IN_{i+2}
ID	224
Source	B
Exist at dest.	No
Next hop	IN_i

Algorithm to test coding feasibility

When selecting messages to code, the switch needs to check the feasibility of decoding the resulting coded message at end nodes. It is possible to decode a message at any end node if the number of unknown packets contained in the coded message is at most one. Moreover, given the characteristic of the XOR operation it is unfeasible to code two messages together if they both contain the same packet with the flag "Exist at Dest." set to false since the packet might vanish from the network without being delivered. Algorithm 5 is performed to check the feasibility of coding messages together.

Coding before transmission

The three packets 223, 224 and 225 are XORed together to form a new coded message. The header of the created message is shown in Table 2.5 and the message is broadcasted into the network. The case of combining two coded messages can easily be deduced and will not be detailed here.

2.3.8 Decoding process

The decoding process can occur at end nodes (centralized approach) or at intermediate nodes (distributed approach). Decoding requires both buffering and computation. Buffers are needed to keep copies of sent and received packets that are needed later in

Algorithm 5 Testing Coding Feasibility

```
1: loop on each packet i in the coded message header
2:     Count ← 0
3:     loop on every packet P in every message to code
4:         if Source of P≠ i and Exist at Dest. = No then
5:             if similar packet with Exist at Dest. = No
                OR
               no similar packet exists then
6:                 Count ← Count + 1
7:             end if
8:         end if
9:     end loop
10:    if Count > 1 then
11:        Set Coding to unfeasible
12:        Stop
13:    end if
14: end loop
15: Set Coding to feasible
```

the decoding process. Decoding itself is not straightforward. The decoding node should XOR the received message with every buffered packet that is part of the coded message. In this thesis, we study the advantages and disadvantages of both approaches.

2.3.9 Simulation results

We ran our simulation on a network of 5 nodes, A, IN_1, IN_2, IN_3 and B. In this network, both sources A and B are exchanging packets at the same rate on a static link. The system is synchronized in the sense that all nodes transmit at the same time. We consider also two channel use on each link. The objective of the simulation is to measure the cardinality (*i.e.* the number of packets) of the coded messages released by different nodes. Figure 2.13 shows the cardinality of coded messages released by A, and the cardinality of coded messages released by IN_1 and IN_2. It is important to note that the cardinality of the coded messages remain low due to two facts, first the XOR operation that eliminates duplicated packets and second, the fact that the network activities are synchronized.

In a second time, the synchronization of the system is removed and the transmission at each node is a stochastic process, where each node has a certain probability to trans-

Table 2.5: Header of created coded message

Unicast	No
Addr1	A
Addr2	B
Coming from	IN_i
Nb. of packets	3
ID	223
Source	A
Exist at dest.	No
Next hop	IN_{i+1}
ID	224
Source	B
Exist at dest.	No
Next hop	IN_{i-1}
ID	225
Source	A
Exist at dest.	Yes
Next hop	IN_{i-1}

mit its coded message. With a probability of 0.75%, Figure 2.14 shows the cardinality of the coded messages at A, IN_1 and IN_2. This simulation shows that the cardinality is continuously increasing and requires further study of the behavior of the system. Note that the behavior of the cardinality remains similar for different transmission probabilities.

2.4 Relay correlated network coding

The address correlated NC can be extended to relay correlated network coding (RCNC). Consider the network illustrated in Figure 2.15 where A is sending a file to B and C is sending a file to D. With address correlated NC, no saving on the bandwidth can occur since no address correlated packets exist among the flow on intermediate nodes n_i; $i = 1$ *to* 4. However, if address correlated NC is applied between relays n_1 and n_4 then a considerable saving on the bandwidth can be made due to the presence of correlation in the flow between n_1 and n_4. The RCNC is not only theoretical. Current networks architecture makes the use of RCNC feasible and practical. Most networks connected to

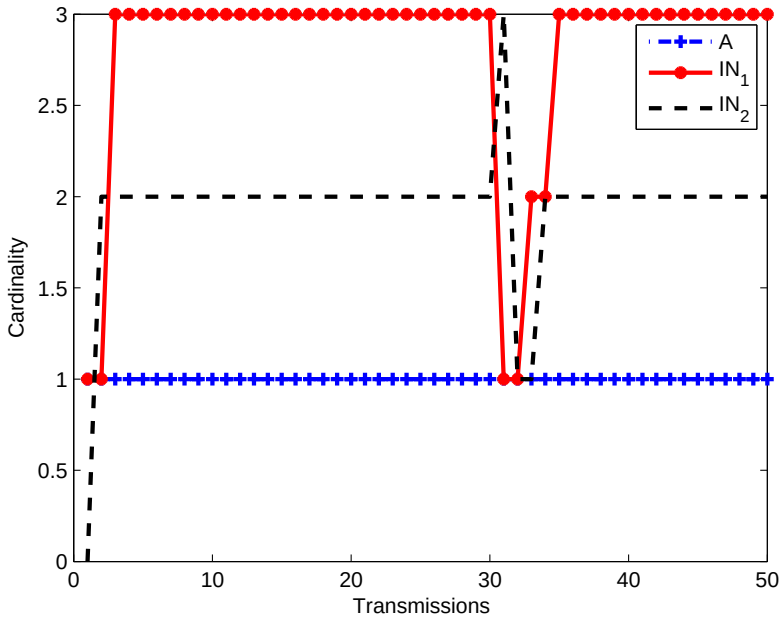


Figure 2.13: Cardinality of coded message at different nodes

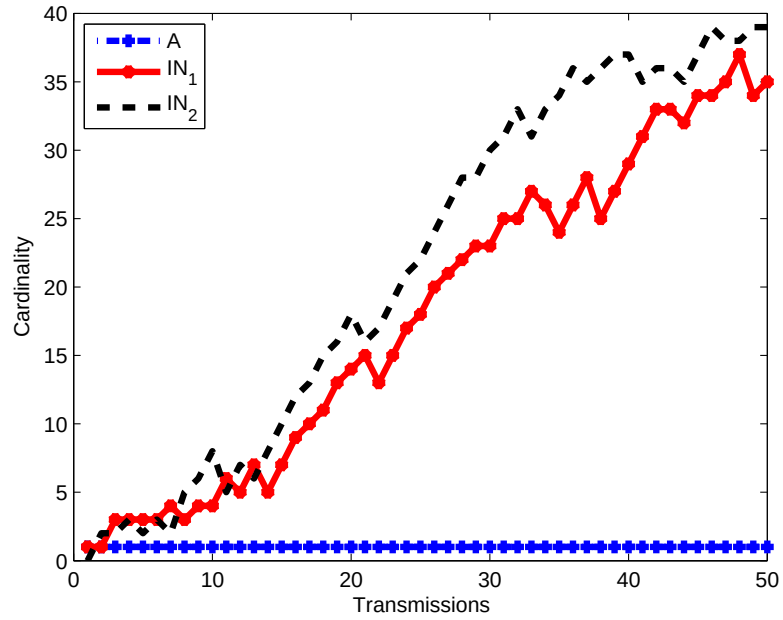


Figure 2.14: Cardinality of coded message at different nodes without synchronization

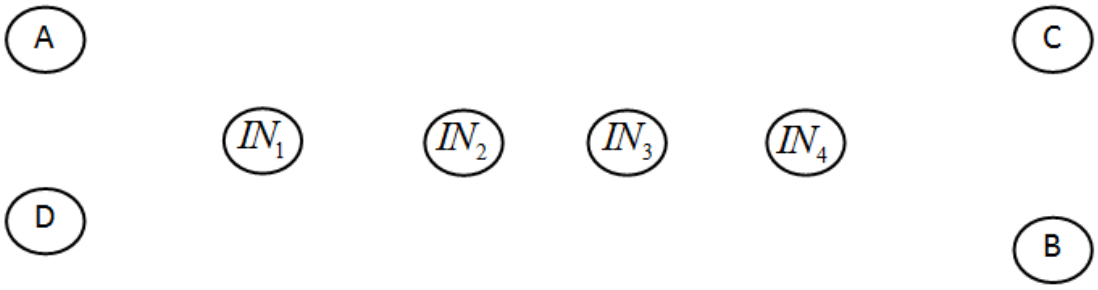


Figure 2.15: Example of networks where RCNC can be used

the Internet are based on the TCP/IP stack where IP version 4 is implemented. Each device on the internet has a unique IP address made of 32 bits. Due to the large number of devices connected to the Internet, a 32 bit field is not enough to connect every device on the globe. IP version 4 offers a solution identified as network address translation or (NAT).

With NAT, all local area networks (LAN) use private IP. The gateway routers perform NAT to translate the private IP to a public IP allowed to travel on the Internet. With this mechanism, the number of possible devices connected to the Internet exceeds the capacity of the 32-bit IP field. With NAT, packets traveling through the public network are seen as packets exchanged between gateway routers. In such case a mass exchange between two networks is relay correlated and RCNC can be applied to save on bandwidth. With IPv6 the concept of public / private IP is not clear. Private IPs exist in IPv6 (prefixed with 0xFD) but NAT translation is not, or at least to be avoided.

2.5 Conclusion

In this chapter, we introduced address correlated NC as a new protocol that enhances bandwidth usage when two nodes are exchanging information. We demonstrated that with the use of adaptive flag, we are able to achieve throughput gain with NC by performing simple algorithms to guarantee decodability at end nodes. With ACNC, intermediate nodes perform only coding operations and route messages to multiple destinations. Contrary to other NC protocols, decoding is only performed at end nodes where new messages are extracted from every received coded message. Consequently,

we are able with ACNC, to route coded messages to their intended destinations using a specific routing mechanism also based on adaptive flag.

We then provided a new packet selection process. For this process, we proposed an optimization model that takes into consideration throughput improvement by forcing the selection of address correlated packets, and delay management by forcing the selection of delayed packets. We then proposed a low cost algorithm based on address correlation to efficiently select packets to be coded and released into the network. The algorithm not only improves bandwidth where the number of received packets is larger than the number of transmitted ones, but it also guarantees a certain level of QoS by managing delays and priority of packets. At last, the study investigated in this chapter has been accepted in two international conferences, IWCMC [C3] and ICCIT [C4].

CHAPTER 3

Centralized decoding

In Chapter 2, we proposed the ACNC algorithm and clarified the coding and the decoding processes. In this chapter, we detail the centralized decoding approach where intermediate nodes are responsible for seizing coding opportunities to code messages together in order to reduce bandwidth utilization, and end nodes are responsible for decoding. This process requires significant buffering capability and increases the computation complexity at end nodes. However, no overhead is added to intermediate nodes where only coding algorithms are implemented.

In this chapter, we study ACNC at both intermediate and end nodes. The contribution of this chapter is of several folds. First, we present a decision model at nodes that balances between the receiving and the sending queues. Second, we present the coding graph, a new technique that helps counting the number of transmissions and coding opportunities when ACNC is used on linear networks. At intermediate nodes, we study the coding opportunities and analyze the size, byte overhead and information contents of coded messages. Third, we study the buffering requirements at end nodes, and the complexity needed for decoding and finally, we set a maturity restriction on packets in the network to prevent packets to live forever in coded messages.

3.1 Decision models for centralized decoding

In this thesis, the considered network consists of nodes that implement a server managing two queues as shown in Figure 3.1; the receiving queue and the outgoing queue. The receiving queue is used to store all received messages and has a capacity denoted by C_{r_q}. We also denote by $x_{r_q}(t)$ the number of messages at time t waiting in the receiving queue. The server investigates the collected messages in the receiving queue to search for coding opportunities on address correlated packets which are only allowed to be coded together. Coded messages are then transferred to the outgoing queue. Therefore, the outgoing queue holds messages ready to be transmitted to the network. We denote by C_{o_q} and $x_{o_q}(t)$ the capacity and the number of coded messages

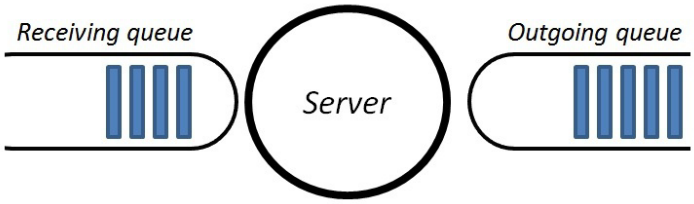


Figure 3.1: Node model

at time t in the outgoing queue respectively. The presence of two queues facilitates the scheduling task at the node and reduces the time needed to search for matching messages to be coded.

The server mainly performs one of the following activities:

- Packet generation denoted by g
- Coding activity denoted by c
- Sending message denoted by s
- Decoding activity denoted by d

The assignment of activities to nodes is completed according to the node role as described in the following scheme:

- Sending node activity set = $\{g, s\}$
- Intermediate node activity set = $\{c, s\}$
- Receiving node activity set = $\{d\}$

Note that, when a network node plays more than one role, its set of activity is the union of the activity set associated to each role. For example, the set of activities at an end node is $\{g, s, d\}$. At time t, each node selects an activity to perform from its activity set. A decision model is needed at each node to select the best activity. Two decision models are adopted for our implementation, a deterministic decision model for intermediate nodes and a probabilistic model for end nodes.

State diagram at intermediate nodes

Intermediate nodes are responsible for coding and sending messages. The state diagram of an intermediate node is shown in Figure 3.2. In this diagram, the coding phase is represented by a link from the receiving queue to the server where two or more messages are selected from the receiving queue to be coded together and sent to the outgoing queue. Note that some messages might be transferred from the receiving to

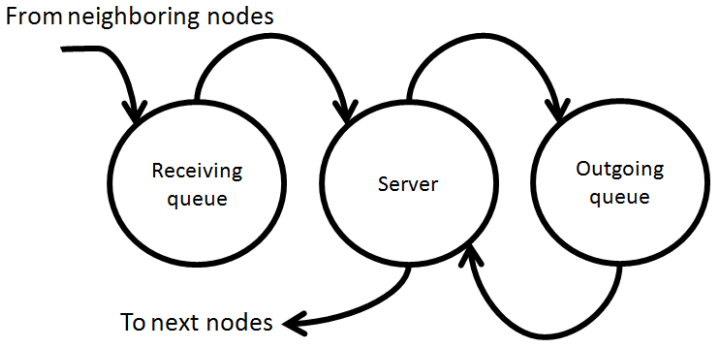


Figure 3.2: Intermediate node state diagram

outgoing queue without being coded, this happens to preserve QoS. The transmission phase is represented by a link from the outgoing queue to the server where one message is transmitted by the server to neighboring nodes. Note that the coding activity includes updating message headers according to the routing table at each node.

State diagram at end nodes

Similarly, the state diagram of an end node is shown in Figure 3.3. In this diagram, the decoding phase is represented by a link between the receiving queue and the server where one message is selected to be decoded and transferred eventually to upper layer. A message containing several packets can be decoded if a maximum of one of these packets does not exist at destination. If the server is unable to decode a message, the message is returned to the receiving queue and further retries are performed later when more received packets are available. The generation phase is represented by a link between the server and the outgoing queue and packets are generated according to a model that will be described later. Finally, the transmission phase is represented by a link from the outgoing queue to the server where one message is transmitted to neighboring nodes.

3.1.1 Deterministic model for intermediate nodes

At intermediate nodes, the decision fluctuates between coding and sending messages and the selection of the activity to be performed largely depends on the number of messages in the queues. The decision model needs to constantly balance the occupation of both receiving and outgoing queues. When balanced, the decision model preserves QoS and favors early received messages in both queues to be processed. For that purpose,

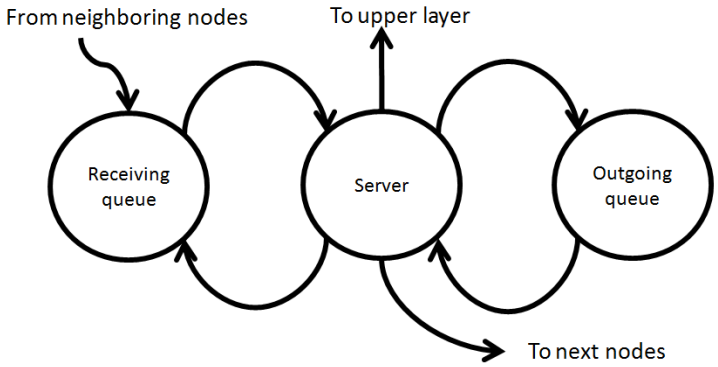


Figure 3.3: End node state diagram

an integer programming problem is proposed where a binary variable y_i, $i \in \{c, s\}$ is assigned to each possible activity. y_i is 0 if the activity is not performed and 1 elsewhere. The model is shown as follows:

$$\max \quad y_c(t) \sum_{i=1}^{x_r(t)} m_{r_i}(t) + y_s(t) \sum_{i=1}^{x_o} m_{o_i}(t) \tag{3.1}$$

$$\text{subject to} \quad y_c(t) + y_s(t) = 1 \tag{3.2}$$

$$y_c(t) \geq \frac{(x_r(t) - x_o(t)) - d_{ro}}{C_r} \tag{3.3}$$

$$y_s(t) \geq \frac{(x_o(t) - x_r(t)) - d_{or}}{C_o} \tag{3.4}$$

$$y_c, y_s \in \{0, 1\} \tag{3.5}$$

In this problem, d_{ro} (respectively d_{or}) is the acceptable difference in number of messages between the receiving and outgoing queues (respectively between the outgoing and receiving queues), and $m_{r_i}(t)$ is the amount of time message i has been in the receiving queue while $m_{o_j}(t)$ is the amount of time message j has been in the outgoing queue.

The constraint (3.3) forces $y_c(t)$ to be 1 whenever the difference between the number of messages waiting to be further coded $x_r(t)$ and the number of messages waiting to be transmitted $x_o(t)$ reaches or exceeds the value d_{r_o}. In this case, the right hand side of (3.3) is strictly positive forcing $y_c(t)$ to take a non zero value. Similar objective can be found for (3.4) by forcing $y_s(t)$ to be 1 when the difference between the number

of messages waiting to be transmitted $x_o(t)$ and the number of messages waiting to be further coded ($x_r(t)$) reaches or exceeds d_{r_o}. The values C_r and C_o are selected to be large enough so that the right hand sides of both (3.3) and (3.4) do not exceed 1. Equation (3.2) guarantees that the server performs exactly one task at a time. It should be noted that (3.3) and (3.4) are built to guarantee that only one of the binary variables $y_c(t)$ and $y_s(t)$ is forced to be 1, however, both variables are allowed be 0 at the same time which explains the need for (3.2).

When the load of both queues is balanced, both binary variables are unrestricted by the constraints of the model and (y_c, y_s) is chosen in order to maximize (3.1). Naturally, the choice forces the release of messages waiting for a longtime in the queues.

3.1.2 Probabilistic model for end nodes

At end nodes, a binary variable y_i, $i \in \{d, g, s\}$ is assigned to each possible activity. y_i is 0 if the activity is not performed and 1 otherwise. As for the case of intermediate nodes, the server can perform exactly one task at a time. We denote by $P(d)$, $P(g)$ and $P(s)$ respectively the decoding, generation and sending probabilities.

The packet generation is considered as an independent event from other activities where the decision of the activity to be undertaken at end nodes is performed in two phases. During the first phase, a decision is taken on whether or not the node should generate a packet. If no packet is to be generated, we move to the second phase to select, according to a stochastic decision model, the node activity to be performed.

The stochastic decision is based on the model proposed in [50] where packets routing is described by probabilistic functions depending on the application. For our case, we use the following equations:

$$P(d) = \frac{e^{-\beta x_{oq}(t)}}{e^{-\beta x_{oq}(t)} + e^{-\beta x_{rq}(t)}} \tag{3.6}$$

$$P(s) = \frac{e^{-\beta x_{rq}(t)}}{e^{-\beta x_{oq}(t)} + e^{-\beta x_{rq}(t)}} \tag{3.7}$$

Where β is a control parameter and $P(d) + P(s) = 1$. In (3.6) decoding probability decreases when the number of messages in the outgoing queue increases favoring in this case the sending activity, and a similar argument can be given to (3.7). However, even when the number of coded messages in the outgoing queue (respectively receiving queue)

exceeds the number of coded messages in the receiving queue (respectively outgoing queue), there is still a non zero probability to select decoding (respectively sending) activity.

As noted in [50], if we set $\beta = 0$, both probabilities are equal to 1/2 and decoding or sending are chosen randomly. On the other hand, if β grows to infinite, $\lim_{\beta\to\infty} P(d) = 0$ and $\lim_{\beta\to\infty} P(s) = 1$ when $x_{oq}(t) > x_{rq}(t)$ thus creating a deterministic decision model where the activity is selected according to queues occupation.

To argue the usage of a probabilistic model at end nodes we consider the following facts:

- The generation of packets is an application activity considered as a random process.
- Contrary to intermediate nodes, end nodes are not completely dedicated for network activities, and the generation of packets is a stochastic process.

3.1.3 Activity selection algorithms

Models presented in Sections 3.1.2 and 3.1.1 are used to propose two low cost algorithms to select the best activity at intermediate nodes and at end nodes respectively. These algorithms are initiated each time the node has to take a decision. The activity selection scheme is detailed in Algorithm 6.

In this algorithm, steps 1 and 3 represent constraints (3.3) and (3.4) in the optimization problem presented in (3.1) and select the activity to be performed depending on the occupation of the receiving and outgoing queues. When queues are almost balanced, steps from 6 to 17 attempt to maximize (3.1) helping late packets to be released from the queues and improving QoS.

The decision model algorithm at end nodes is presented in Algorithm 7. In this algorithm, step 2 independently selects the generation activity with a certain predefined probability. If no new packets are generated, the code between lines 5 and 10 gives higher probability to the activity that balances the receiving and outgoing queues.

3.2 Study of coding and decoding activities

In this section, we closely study the centralized decoding from different points of view. Initially, we plot the coding activities through intermediate nodes and investigate

Algorithm 6 Activity selection algorithm: Intermediate nodes

if $\frac{(x_r(t)-x_o(t))-d_{ro}}{C_r} > 0$ **then**
    return Coding
else if $\frac{(x_o(t)-x_r(t))-d_{or}}{C_o} > 0$ **then**
    return Sending
else
    $delay_r = 0$
    for $i = 1$ to $x_r(t)$ **do**
        $delay_r = delay_r +$ (t - arrival time of i)
    end for
    $delay_o = 0$
    for $i = 1$ to $x_o(t)$ **do**
        $delay_o = delay_o +$ (t - arrival time of i)
    end for
    if $delay_r > delay_o$ **then**
        return Coding
    else
        return Sending
    end if
end if

coding opportunities at the nodes and message cardinality in the network. We also study the decoding opportunities and packets buffering at end nodes.

In a synchronized environment with period T, Figure 3.4 shows coding activities in a network with 2 intermediate nodes. Each arrow is unicast transmission while dashed arrow is a broadcast taking place after each coding activity. End nodes generate and transmit a packet at the beginning of each period. We assume two channel uses on each link where a node can send and receive packets at the starting of each period. The figure shows the increases of the cardinality of the coded message at each period. At the end of iteration 6 for instance, each coded message in the network holds up to 5 packets.

Figure 3.5 shows a similar trace in a 5 nodes network. However, in this network, the absorbing effect can be observed. The absorbing effect emerges when more than two intermediate nodes exist in the network, leading to a reduction of coded message cardinality. Indeed, if a packet is contained by two coded messages received by intermediate node, this packet vanishes from the resulting coded message, due to the property of XOR since $a \oplus a = 0$ and $b \oplus 0 = b$. This process creates an absorbing effect and hence reduces

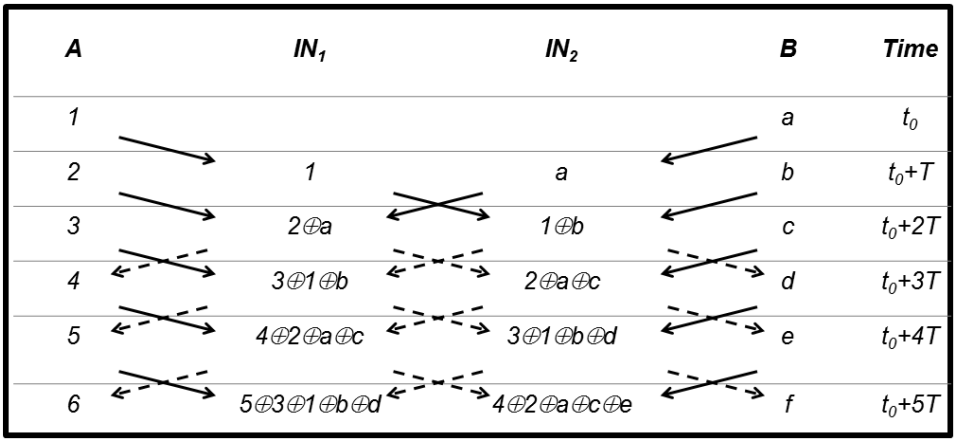


Figure 3.4: Coding activities in a 4 nodes network

the cardinality of coded messages. However, there is no guarantee that the architecture of the network will always favor a complete synchronization of the network activities, thus no guarantee that the network will always favor the absorbing effect.

3.2.1 Coding graph

To analytically compute the cardinality of coded messages in the network, we introduce the coding graph shown in Figure 3.6 and Figure 3.7 with odd and even number of intermediate nodes respectively. Note that, in these graphs, it is assumed that the network is synchronized with a time period T. Coding graphs demonstrate the coding opportunity when packets are exchanged between end nodes. After coding, coded

Algorithm 7 Activity selection algorithm: End nodes

1: $P(\text{generation}) = random(0, 1)$
2: **if** $P(\text{generation}) <= (\text{Generation-threshold})$ **then**
3: return Generation
4: **end if**
5: $P(\text{decoding}) = \frac{e^{-\beta x_{oq}(t)}}{e^{-\beta x_{oq}(t)} + e^{-\beta x_{rq}(t)}}$
6: **if** $P(\text{decoding}) <= random(0, 1)$ **then**
7: return Decoding
8: **else**
9: return Sending
10: **end if**

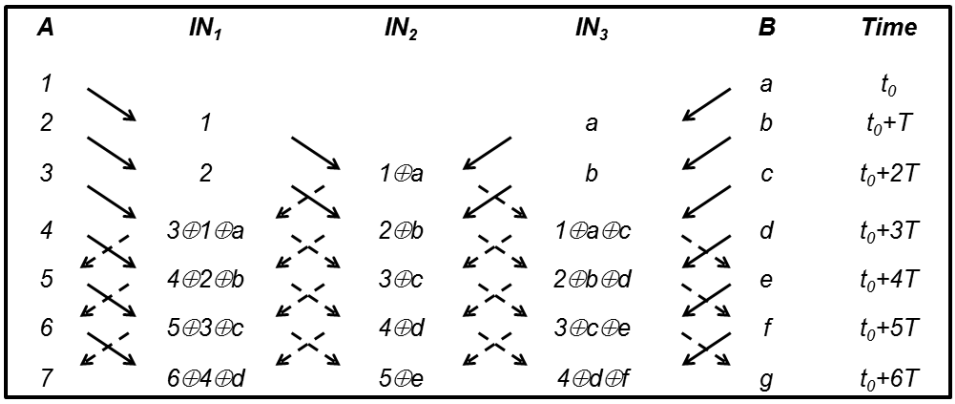


Figure 3.5: Tracing coding activities in a 5 nodes network

messages are broadcasted and received by the two neighboring nodes.

Arrows in the coding graph represent newly generated packets by end nodes. In Figure 3.6 the first coding opportunity occurs at iteration 4 (top level of the graph) when the first two generated packets meet at IN_4 creating the first coded message with cardinality of 2. While this coded message is broadcasted to neighboring nodes IN_3 and IN_5, intermediate node IN_4 receives a new set of packets creating a second coded message with cardinality of 2. Black and gray vertices show coding opportunities with newly created packets while white vertices show coding opportunities between coded messages. In each of these coding graphs, two coded messages, shown in two different colors (*i.e.* black and gray), are constructed and exchanged by intermediate nodes. After a transition time, these coded messages reach the end nodes and start delivering packets. We notice in these figures that at each iteration, two packets are released from the two end nodes while an average of one packet is injected into each coded message. In fact, when the delivery process started, Figure 3.6 shows two packets added to the black coded message every two iterations thus an average of one packet per iteration. The same occurs for the gray coded message. These coding graphs are used by Theorem 2 to compute the cardinality of coded messages in the network.

Theorem 2. *Given a linear network $G = (N, E)$ with two end nodes and P packets exchanged between end nodes ($P/2$ released by each end node) an upper bound on coded message cardinality at a given iteration i at any intermediate node in the network is*

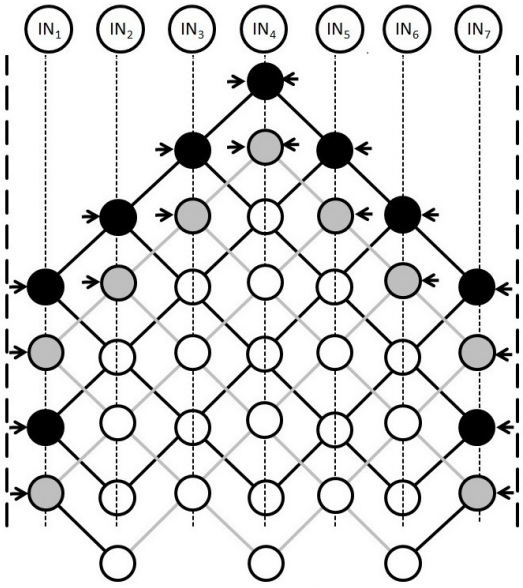


Figure 3.6: Coding graph for 7 intermediate nodes (odd number of nodes)

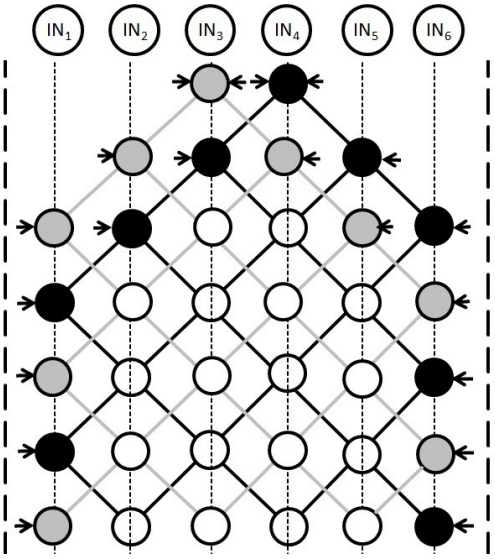


Figure 3.7: Coding graph for 6 intermediate nodes (even number of nodes)

given by:

$$C(P,N,i) \leq 2 \min\left(i, \left\lceil \frac{N-2}{2} \right\rceil\right) + \max\left(\left(\min\left(i, \frac{P}{2}\right) - \left\lceil \frac{N-2}{2} \right\rceil\right), 0\right) \quad (3.8)$$

where N is the number of nodes in the linear network.

Proof. The first part of (3.8) counts the number of new packets added to the coded message at each iteration while the message is not yet delivered to the end nodes *i.e.*, still within the vertical bold dashed lines. In this case, the message receives two new packets at each iteration. The number of iterations before delivering the message to an end node is $\lceil \frac{N-2}{2} \rceil$ which is the distance between the central intermediate node IN_4 and the end nodes according to Figure 3.6. The second part counts the number of new packets added to the message after being delivered to end nodes i.e, after $i > \lceil \frac{N-2}{2} \rceil$. In this case and referring also to Figure 3.6, two packets are added to the coded message each two iterations.

The same proof applies when the number of intermediate nodes is even. In fact, a simple linear translation of the gray nodes in Figure 3.7 leads to the structure of the coded graph in Figure 3.6. ■

Simulations are done to experimentally compute the cardinality of coded messages within different types of networks and at different intermediate nodes. As the size of a coded message can grow infinitely depending on the number of packets exchanged, simulations are conducted on networks with an even and odd number of nodes respectively. 1000 packets (500 packets transmitted from each end node) are exchanged between the two end nodes and results in Figure 3.8 and Figure 3.9 show the growth of the cardinality of coded messages transmitted from selected nodes. The figures show the upper bound on the cardinality of the coded messages given in Theorem 2.

3.2.2 Buffering time

The buffering time of a packet at a node is computed as the difference between the time the packet has started being stored at the node and the last time the packet is used for decoding. Figure 3.10 shows in percentage with respect to the total communication time, the highest and the average buffering time at each node of the network. As shown in the figure, the buffer size at each of the end nodes should be large enough to hold all exchanged packets since the largest buffering time is almost equal to the total transmission time needed to deliver all the packets. This is justified by the fact that the

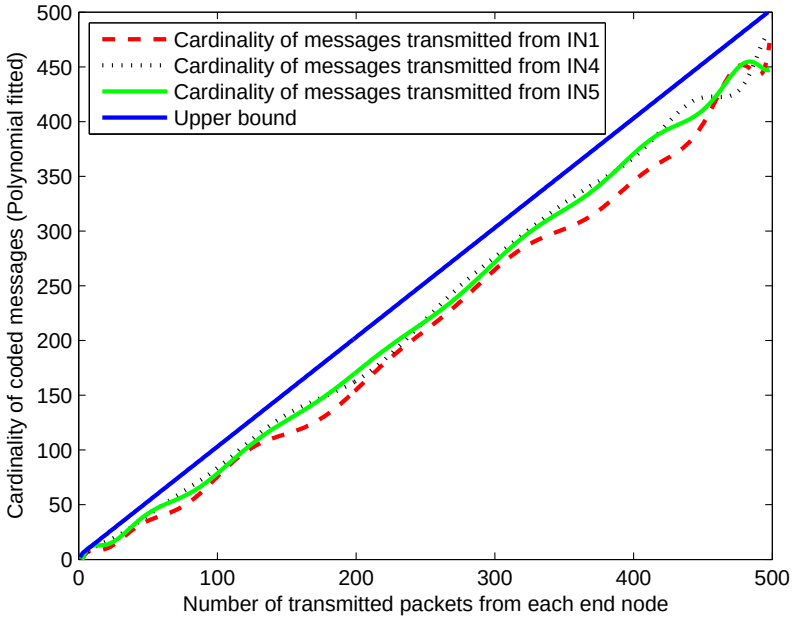


Figure 3.8: Cardinality of coded messages with centralized decoding for a network of 8 nodes

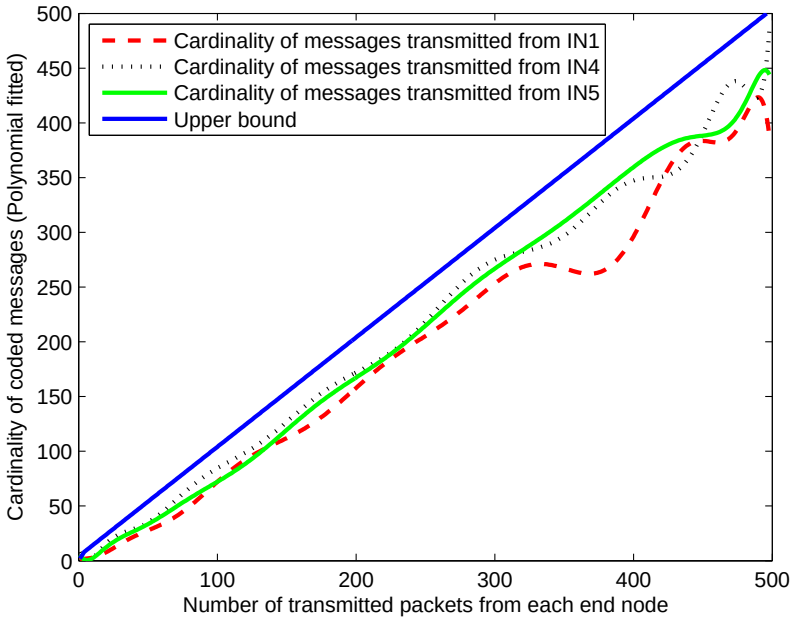


Figure 3.9: Cardinality of coded messages with centralized decoding for a network of 9 nodes

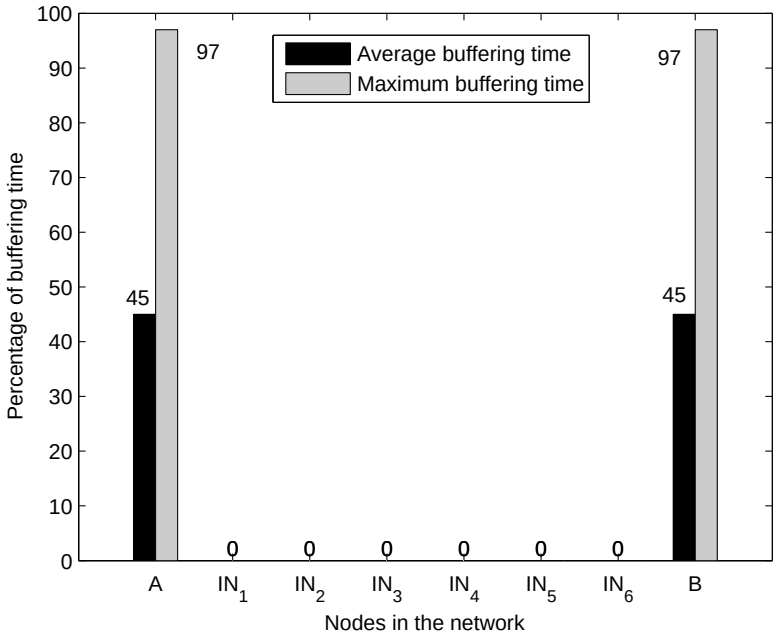


Figure 3.10: Decoding at end nodes: buffering time at each end node (1000 packets exchanged in network of 8 nodes)

early generated packets might remain as part of the coded messages till the completion of the communication.

Figure 3.11 shows a polynomial fit of the buffering time in seconds of the received packets at each of the end nodes. The total duration of the simulation is 5.7 seconds. This figure completes Figure 3.10 which indicates that early transmitted packets remain part of the coded messages till the end of the communication. These packets create an overhead in the coded message since they are already delivered and exhibit complexity and buffering overhead at end nodes. In Section 3.3, the concept of aging is introduced to remediate to this problem.

3.2.3 Byte overhead transmission

Without NC, exactly $(N-1)P$ transmissions are needed to exchange P packets between the two end nodes of a linear network. With NC, this number is dramatically reduced as noticed in the coding graphs. A lower bound on the number of transmissions

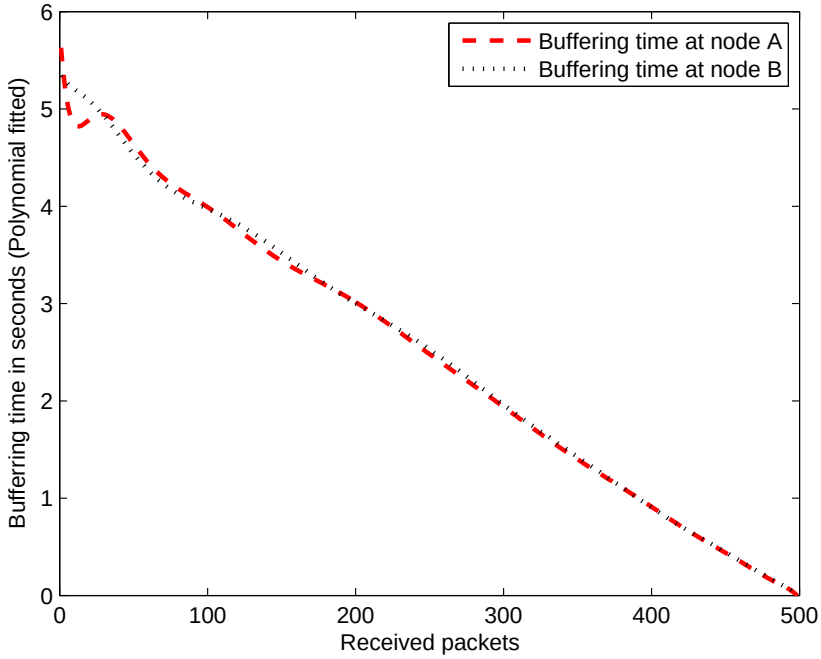


Figure 3.11: Buffering time of received packets

needed to exchange P packets is

$$P + (N-2)\frac{P}{2}$$

where the first term accounts for the transmissions at the 2 end nodes and the second term accounts for the transmissions at the intermediate nodes when considering the optimal case where each intermediate node is continuously performing NC operation saving one transmission over 2. Thus, a lower bound to the total number of transmissions equals to $N \times \frac{P}{2}$.

Table 3.1: Lower bound on the number of bytes exchanged with and without NC

Protocol	Transmissions	Exchanged bytes	
Traditional store and forward	$(N-1)P$	$(N-1)PP_a$	(3.9)
NC-decoding at end nodes	$N\frac{P}{2}$	$N\frac{P}{2}P_a + P_h(N-2)\frac{P(P+2)}{8}$	(3.10)

Denoting by P_a the size in bytes of a packet and by P_h the size of the packet header, the total number of bytes exchanged with and without NC is presented in Table 3.1.

With NC where decoding is conducted at end nodes, the cardinality of the coded messages increases linearly with the number of packets transmitted, leading to an increase in the number of bytes exchanged. This increase is due to the overhead exhibited in the header where bytes are added for each packet taking part in a coded message. Referring to Theorem 2, the size of the headers generated by the NC process is computed in bytes as $(N-2) \times P_h \times \sum_{i=1}^{\frac{P}{2}} i$ which can also be written as $(N-2) \times P_h \times \frac{P(P+2)}{8}$.

3.3 The aging concept

We consider again linear networks where end nodes exchange information synchronously with a time period T and two channels are available on each link between the nodes of the network allowing simultaneous transmissions of two messages during each time period T. The network model is represented as an undirected graph $G = (N, E)$ where the set of nodes N is divided into two subsets, $\{A, B\}$, the set of end nodes and IN the set of intermediate nodes.

3.3.1 Definitions and principles

The following definitions are needed in order to describe the aging concept.

Definition 16. *The age of a packet is defined as the number of times the packet contributes to a coded message and the age of a coded message is defined as the age of the oldest packet in the coded message.*

Definition 17. *The coding capacity of a network, denoted by Ç is defined as the number of coded or uncoded messages that can be further coded together.*

A coding capacity of two indicates that any released coded message from a node is the resulting of coding at most two received messages together.

Definition 18. *The diversity of a coded message is defined as the number of distinct packets within the coded message.*

Note that, when XORing is used to code messages together, diversity and cardinality coincide.

Definition 19. *The maturity of a network is defined as the highest allowed age for all the packets in the network.*

To prevent packet i from growing in age beyond maturity, we prevent coding operations with messages containing i. For each newly generated packet i, we assign the aging variable α_i initially set to 0. Each time a coded message is created with packet i, the aging parameter of that packet is incremented by 1. The age of a coded message C_j, denoted by α_{C_j}, is calculated as follows: $\alpha_{C_j} = \max(\alpha_i) \; \forall \; i \; \in \; C_j$. Two parameters are set by the network and used by intermediate nodes when messages are coded together: ζ denoting the coding capacity of the nodes and μ denoting the maturity of the packets.

3.3.2 Cardinality of messages in NC

In some communication networks such as satellite networks, a single intermediate node (relay) is used to exchange information between end nodes. The intermediate node codes received packets and broadcasts the coded messages back to source nodes. A simple XORing of the received messages is enough to double the bandwidth and increase the transmission rate [51, 52]. When the number of intermediate nodes exceeds one, coding becomes more challenging. In this context, intermediate nodes receive not only newly generated packets but also coded messages that need to be further coded together.

Let us consider the network of Figure 3.4. End node A generates packets labeled 1,2,3,4... While end node B generates packets labeled a, b, c, d... Intermediate nodes, *i.e.* IN_1 and IN_2 process the received packets and broadcast coded messages to neighboring nodes. Notice that packet a for example is oscillating between two intermediate nodes within a coded message and the cardinality of the coded message increases by one at each iteration. The decoding of the received messages at end node A or end node B requires the complete buffering of all already transmitted and received packets. When a network contains more than two intermediate nodes, the absorbing effect emerges leading to a reduction in the cardinality of coded messages. However as stated in Section 3.2, there is no guarantee that the architecture of the network will always favor the absorbing effect.

3.3.3 Aging and maturity in network coding

To prevent the cardinality of coded messages from increasing indefinitely and to make sure that packets released into the network will vanish after a certain amount of time, we assign an age to each packet, initially set to 0 when the packet is generated and released by an end node. Moreover, the age of a packet is incremented each time

the packet is used as a part of a newly coded message. When a node IN_i receives coded messages, it searches for matching possibilities for further coding by performing Algorithm 8. The aging concept is shown in steps 3 and 7 to ensure that coding is prohibited when needed. The computation overhead created by the aging concept in Algorithm 8 is hence negligible.

Algorithm 8 ValidateAndCode

```
1: for each candidate coded message do
2:     for each packet in the coded message do
3:         if age ≥ mature then
4:             Remove message from candidate list
5:         end if
6:     end for
7: end for
8: Code candidate messages
9: for each packet in the resulting coded message do
10:    Increment the age
11: end for
```

A packet can be part of new coded message until being mature. A coded message containing a mature packet cannot be further coded with any other packet or coded message and should be on its way to vanish from the network.

The next theorem gives an upper bound on the cardinality of coded messages.

Theorem 3. *Given a network $G = (N, E)$, a coding capacity $Ç$ and a maturity scalar μ in $\mathbb{N}^+$, the cardinality of any coded message in the network is bounded by $Ç^\mu$.*

Proof. The proof can be done by induction as follows: let n be the coding iteration at a node, for $n = 1$ a maximum of $Ç$ packets are coded together and the cardinality of the resulting coded message is $\leq Ç$ and an age of 1. For $n = k$, a maximum of $Ç$ coded messages each of cardinality $\leq Ç^{k-1}$ are coded and the cardinality of the resulting coded message is $\leq Ç^k$ and an age of k. The general form of the induction is given by $|C| \leq Ç^n$ and the age of the coded message is n. When a coded message reaches the maturity age μ, the cardinality of the coded message is bounded by $Ç^\mu$ and no further coding can be conducted with the message. ■

The aging concept is illustrated in Figure 3.12. In this example, the maturity age is 2, the coding capacity is 2 and each packet is superscripted by its age. At time $t_0 + 4T$, intermediate node IN_2 receiving $3^1 \oplus 1^2 \oplus b^2$ and d^0 performs Algorithm 8 and

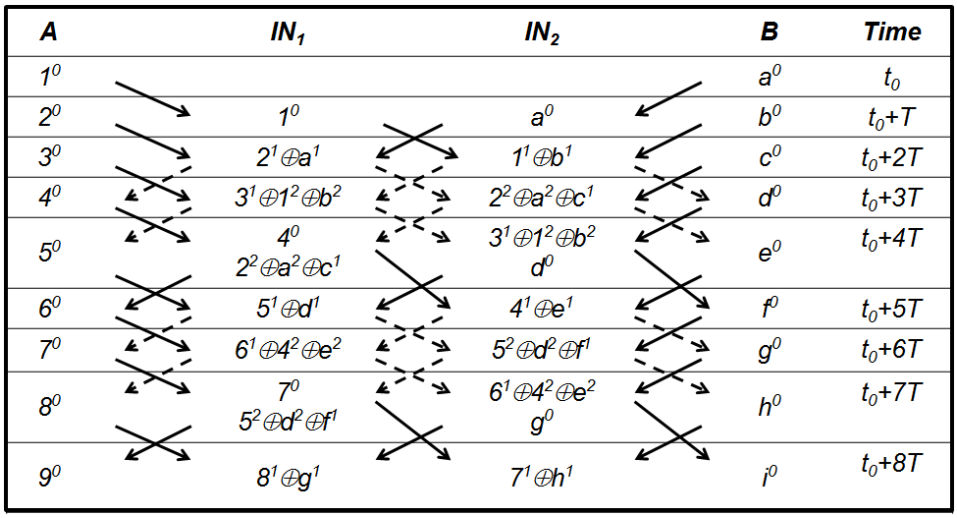


Figure 3.12: The aging effect with two intermediate nodes

decides not to code those two messages together. Note that this will have an effect on the throughput gain since we have now to transmit two coded messages instead of one, because d and 3 need to be delivered to A and B respectively.

As proven by Theorem 3, the selection of the maturity age depends on several criteria that affect the buffering size at the end nodes and the throughput gain as follows:

- The higher the maturity age, the larger should be the end node buffer size since packets can live longer.
- For the same number of packets exchanged, the throughput gain is inversely proportional to the total number of transmissions needed to deliver all the packets.

To compute the number of transmissions needed to exchange packets between end nodes in a linear network, we refer to the coding graph of Figure 3.13 where each node is a coding activity (CA) creating a coded message broadcasted to the two neighboring nodes. Nodes in the coding graph are divided into three groups, *i.e.* the outer nodes, shown in black that perform CA with newly created packets, the inner nodes, in white, that perform CA between coded messages and the gray nodes representing extra nodes needed for convenience purpose to complete the graph. The width of the coding graph, represented by the vertical dash lines, is limited by the number of intermediate nodes. A participating packet (PA) is defined as a packet leading to a CA at a node. A packet with an age equal to μ cannot lead to a CA. Figure 3.13 shows a coding graph with 7 intermediate nodes and maturity $\mu = 7$. In this figure, 10 PA, shown as arrows, are

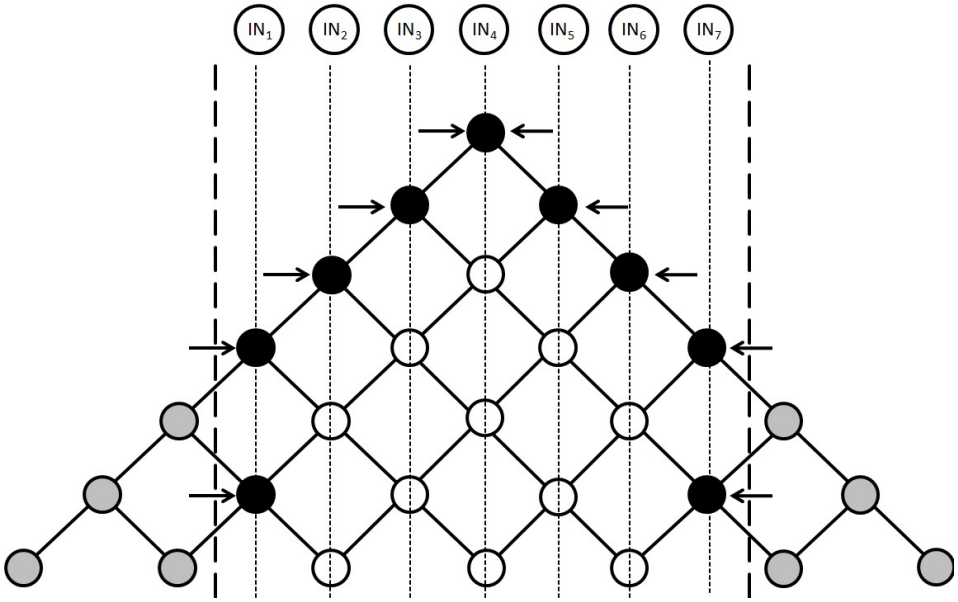


Figure 3.13: A coding graph for a network of 7 intermediate nodes and $\mu = 7$

creating 20 CA, *i.e.* number of nodes between the vertical dashed lines, thus saving 20 transmissions. The generalization of the above example is given in Theorem 4 as follows:

Theorem 4. *Given a linear network $G = (N, E)$ with two end nodes and a maturity scalar $\mu \in \mathbb{N}^+$, the number of transmissions needed to exchange a total of P packets between the two end nodes is given by*

$$T(N,\mu,P) = (N-1)P - P\frac{\sum_{i=1}^{\mu} i - 2\sum_{i=1}^{\mu} \max\left(0, \left\lceil \frac{i-\lceil \frac{N-2}{2} \rceil}{2} \right\rceil\right)}{2\ \min\left(\mu, \lceil \frac{N-2}{2} \rceil\right) + 2\ \max\left(0, \left\lceil \frac{\mu-\lceil \frac{N-2}{2} \rceil+1}{2} \right\rceil\right)} \tag{3.11}$$

Proof. The total number of transmissions T equals to the total number of transmissions without NC, *i.e.* $(N-1)P$, minus the number of transmissions saved thanks to NC. Since each coding activity saves one transmission, the number of transmissions saved is equal to the ratio between the number of coding activities, i.e $CA(N,\mu)$ over the number of participating packets, *i.e.* $P(N,\mu)$, multiplied by the total number of exchanged packets. The number of coding activities, computed from Fig. 3.13, is given by:

$$CA(N,\mu) = \sum_{i=1}^{\mu} i - 2\sum_{i=1}^{\mu} \max\left(0, \left\lceil \frac{i - \lceil \frac{N-2}{2} \rceil}{2} \right\rceil\right) \tag{3.12}$$

where the first term equals to the total number of nodes in the coding graph while the second term equals to the number of extra nodes on both sides of the coding graph. The number of PA, *i.e.* $P(N,\mu)$, is given by

$$P(N,\mu) = 2\ \min\left(\mu, \left\lceil \frac{N-2}{2} \right\rceil\right) + 2\ \max\left(0, \left\lceil \frac{\mu - \left\lceil \frac{N-2}{2} \right\rceil + 1}{2} \right\rceil\right) \tag{3.13}$$

where the first term equals to the number of participating packets in the upper part of the coding graph, *i.e.* before it exceeds the bold dashed lines, while the second term equals to the number of participating packets in the lower part of the coding graph. For P packets exchanged between end nodes the total number of saved transmissions equals $P\frac{CA(N,\mu)}{P(N,\mu)}$, and the proof is complete. ■

3.4 Simulation results

We run a simulation on a network with 5 intermediate nodes and 2 end nodes exchanging information. The simulation is conducted with and without aging. We use a simple model where end nodes generate new packets with a fixed generation rate and intermediate nodes perform Algorithm 8 to code and release messages. In our simulations, the coding capacity parameter Ç is set to 2. Figure 3.14 shows the growth of the cardinality of the coded messages generated by an intermediate node of the network over 3 different runs without the concept of aging. The absorbing effect is visible each time the cardinality of the coded messages is reduced. However it is clear that the trend is towards an infinite growth of the cardinality in time.

Using the same network, Figure 3.15 shows a polynomial fit of the cardinality of coded messages in the queue of an intermediate node when using the concept of aging with different values of maturity, *i.e.* $\mu = 4, 5, 6$. As proven in Theorem 3, the maximum cardinality of coded messages is bounded by $Ç^{\mu}$, *i.e.* 16, 32 and 64 respectively. This bound is never reached as it can be observed in Figure 3.15 where

Fig. 3.16 shows, for different maturity, the total number of transmissions needed to deliver $P = 1000$ packets (500 transmitted from each end node) exchanged between end nodes of a linear network with 7 and 9 nodes. For both cases, this figure gives simulation and analytical results according to Theorem 4. The total number of transmitted packets in the network decreases when maturity increases. With a maturity age $\mu = 0$, no

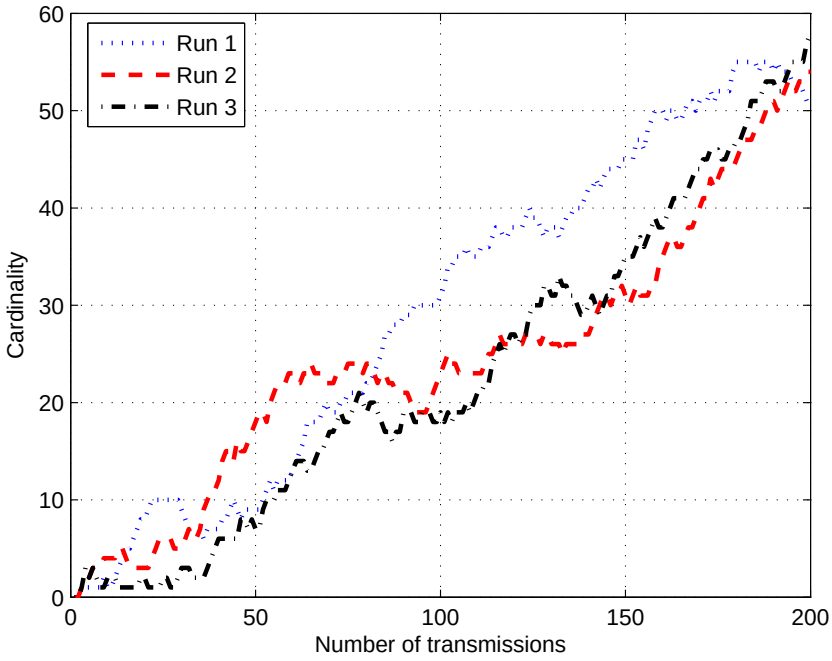


Figure 3.14: Cardinality without aging (network with 7 nodes)

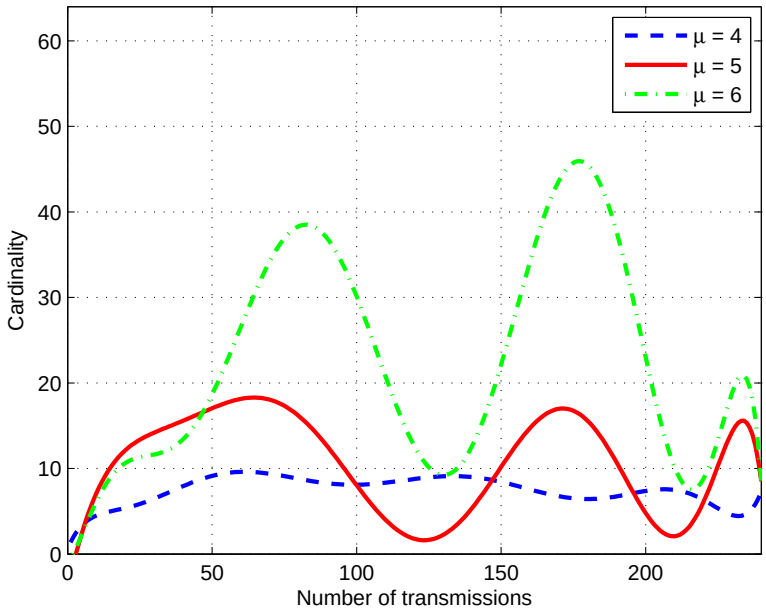


Figure 3.15: Cardinality with aging (network with 7 nodes)

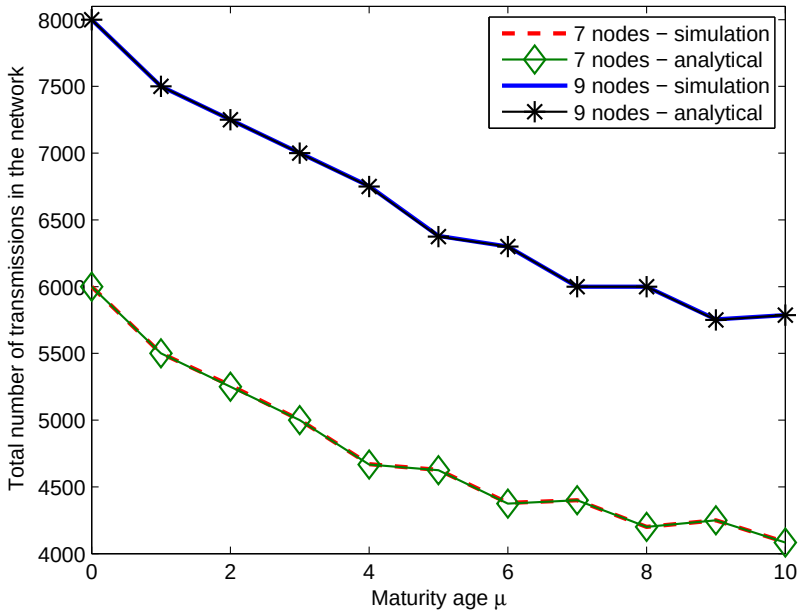


Figure 3.16: Number of transmissions as a function of maturity μ

coding is allowed in the network and P packets require $T = (N-1)P = 6000$ and 8000 transmissions respectively in networks with 7 and 9 nodes. On the other hand when maturity principle is not applied ($\mu \to \infty$) the total number of transmissions is $T = \frac{N-1}{2}P$. Because nodes are continuously allowed to perform NC on packets and messages, both analytical and simulation results match perfectly.

Finally, Figure 3.17 shows the variation of the required average and maximum buffering at end node for different maturity values. As observed earlier, the figure shows that the buffering requirement increases with maturity since packets live longer in the network.

3.5 Conclusion

In this chapter, we showed the importance of aging in NC and the mechanism preventing packets from living infinitely in the network. With aging, the achievable saving in bandwidth utilization is reduced compared to NC without aging. However, the decoding complexity is dramatically reduced by reducing the amount of buffering size at

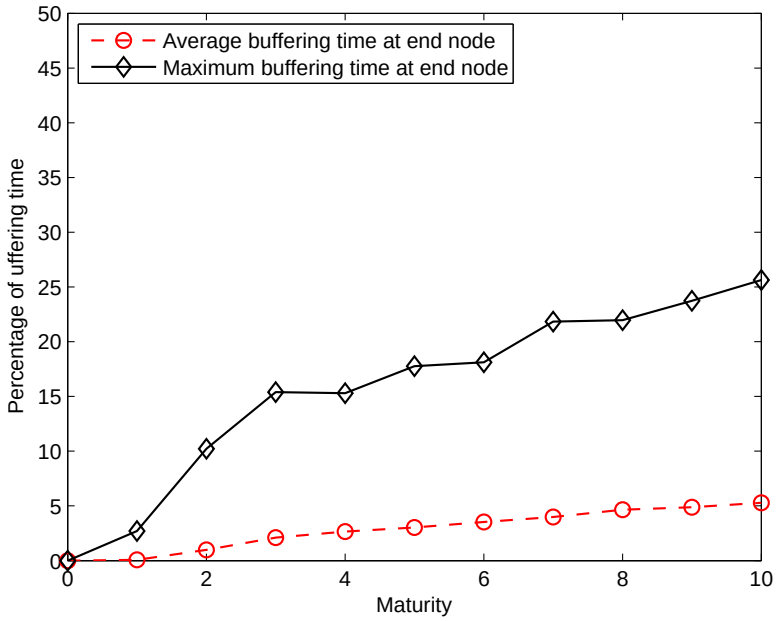


Figure 3.17: Decoding at end nodes: variation of the buffering time for different maturity

decoding nodes. The maturity age should be set in a way to balance buffering size and bandwidth gain according to the buffering capability of end nodes in the network. We believe that the aging concept is a valuable technique to enable the development of NC in practical situations and could be implemented in future network protocols. The same mechanisms presented in this chapter are used and enhanced in chapter 5 where networks are considered to be lossy. At last, the concept of aging investigated in this chapter has been published in IEEE wireless communication letters [A2] while decision models and the coding graphs have been part of the journal paper submitted to EURASIP [A1].

CHAPTER

4 Distributed decoding

In this chapter, we introduce decoding at intermediate nodes where coding is conducted on address correlated packets. In particular, we propose a novel algorithm that enhances the bandwidth usage and reduces the resource allocation by allowing intermediate nodes to perform distributed decoding. Distributed decoding relies on buffering packets at intermediate nodes to reduce the cardinality of transmitted coded messages.

Several NC protocols rely on buffering received packets to forward linear combinations of these packets. The optimality of NC with buffers has been studied in [53]. In their work the authors have analyzed distributed and packetized implementations of random linear network coding and have given formal proof that PNC allows to keep minimal buffer sizes while maintaining optimal performance in terms of bandwidth utilization. In [54], the authors have presented a cross-layer approach referred to Buffer-Aware NC, which allows the transmission of some packets without NC to reduce the packet delay. NC is also introduced in many distributed systems; these systems are characterized by their ability to share resources and the possibility to work in a decentralized environment. A NC algorithm has been proposed in [37] to store data reliably over long period of time using a distributed collection of storage nodes which may be individually unreliable. However, NC itself is inherently distributed by the nature of the encoding schemes where packets injected into the network by end nodes are duplicated by mean of coding and broadcasting. Indeed, this process makes the packet redundancy difficult to control and requires nodes to manage this redundancy in an intelligent way that guarantees decodability and bandwidth usage at the same time [55]. In this chapter, we investigate the packet multiplicity problem and propose a new distributed scheme that eliminates redundant packets without affecting decodability of coded messages.

The contribution of this chapter is twofold; i) creation of a novel distributed decoding mechanism for NC protocol where resource utilization is reduced and distributed among nodes of the network, ii) study of the distributed decoding algorithm efficiency on the total number of transmitted bytes in the network and the buffering time needed at

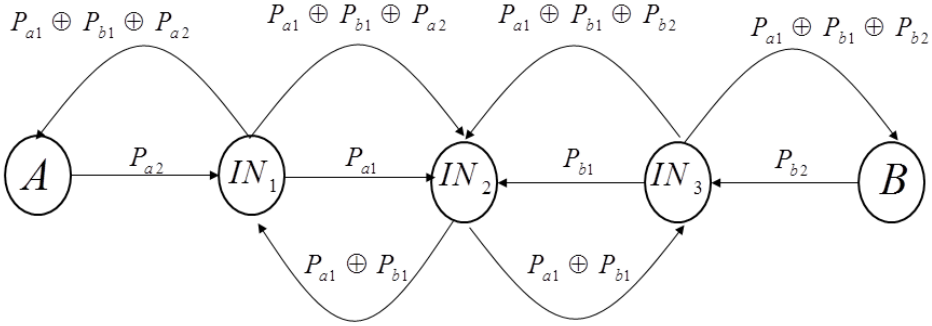


Figure 4.1: NC technique

each node to guarantee decodability of transmitted packets. The chapter is organized as follows; we first present the distributed decoding concept. Then the cardinality of coded messages, the byte overhead and the buffering time are studied under the distributed decoding approach. Performance and simulation results are then given to show the advantages of distributed decoding in terms of buffering time and total number of transmitted bytes.

4.1 Distributed decoding concept

First consider the NC example shown in Figure 4.1 where end node A generates packets with IDs P_{a1}, P_{a2}... while end node B generates packets with IDs P_{b1}, P_{b2}.... The first coding opportunity is detected at IN_2 when the node receives packets P_{a1} and P_{b1}. Node IN_2 creates the coded message $P_{a1} \oplus P_{b1}$ and broadcasts the created message to both neighboring nodes. IN_1 sees another coding opportunity by receiving packet P_{a2} and coded message $P_{a1} \oplus P_{b1}$. It then creates $P_{a1} \oplus P_{b1} \oplus P_{a2}$ and broadcasts the created message. When receiving $P_{a1} \oplus P_{b1} \oplus P_{a2}$, node A can easily extract the new packet P_{b1}.

With the distributed decoding approach, intermediate nodes are also responsible for removing unneeded redundancy in the system without compromising the decodability at the end nodes. In the example of Figure 4.1 where node IN_1 creates coded message $P_{a1} \oplus P_{b1} \oplus P_{a2}$, packet P_{a1} has already been transmitted by IN_1 at a previous iteration and is now in its way to destination. This packet can be safely removed from the coded message provided that P_{a1} has been stored in IN_1.

To better understand the distributed decoding concept, we consider a network with

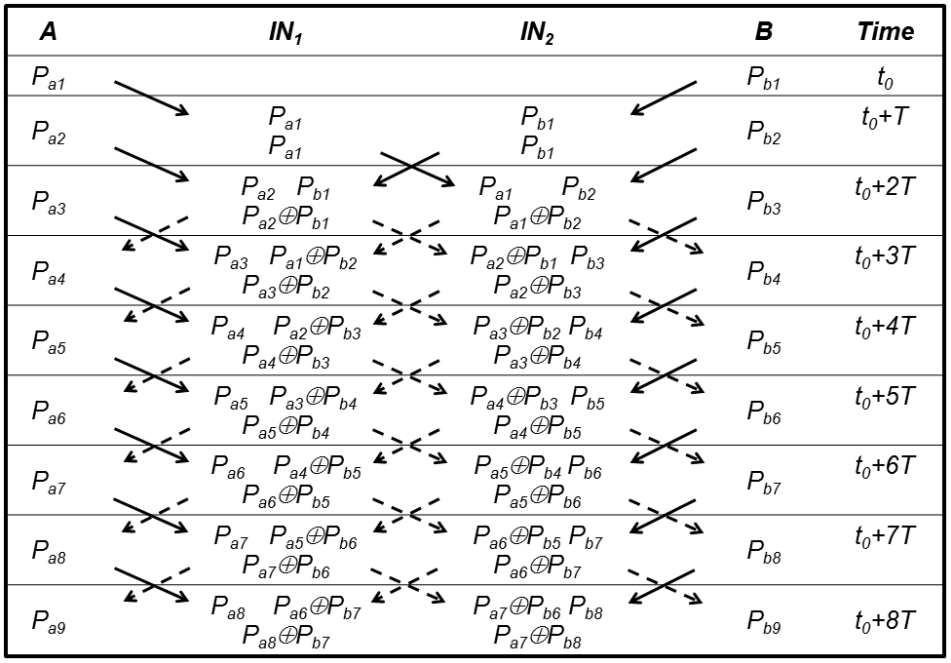


Figure 4.2: Distributed decoding with 2 intermediate nodes

two intermediate nodes and the distributed decoding process illustrated in Figure 4.2. In this figure and for each intermediate node, the received and the transmitted messages are shown at each iteration. At iteration 2 for example (*i.e.* at time $t_0 + 2T$) node IN_1 receives in unicast packets P_{a2} and P_{b1} and creates a coded message $P_{a2} \oplus P_{b1}$. IN_1 also buffers packets P_{a2} and P_{b1} for further decoding. Packet P_{a2} is received back at iteration 4 within the coded message $P_{a2} \oplus P_{b3}$. Being also delivered to destination node B with the same broadcast, packet P_{a2} is redundant and can safely be removed from the coded message.

With distributed decoding, each intermediate node maintains a buffer B which is used to store received packets. When receiving a message, each intermediate node performs Algorithm 9. The algorithm starts by removing, from the received message, packets that are already buffered in the node, thus reducing its cardinality. If at the end of the algorithm the cardinality of the received message is reduced to 1, the only packet remaining is saved in the node's buffer.

The idea behind distributed decoding is as follows: if a node receives a coded message containing a buffered packet, this indicates that the packet already reached this node at

Algorithm 9 Decoding at intermediate nodes

```
for each packet P_i in the received message do
    for each packet P_j in the buffer B do
        if P_i = P_j then
            message = message ⊕ P_j
            Remove from the message the header of P_i
        end if
    end for
end for
if cardinality of the message is 1 then
    add packet to B
end if
```

a previous transmission and it is now on its way to the destination. The presence of this packet in the coded message is the consequence of NC. Distributed decoding is intended to remove the duplicities of such packets in order to reduce the cardinality of coded messages. In the following subsections, we analyze the distributed decoding mechanism from three points of view: first, we outline the cardinality of coded messages; in second place, we examine the buffering time needed at each node; finally, we calculate the total number of bytes exchanged by end nodes.

4.1.1 Cardinality of coded messages

To check the effect of distributed decoding on the message cardinality, simulations were run on a network of 8 nodes and the cardinality of coded messages is shown in Figure 4.3 for three different nodes where we observe that the cardinality varies between 1 and 2. In fact, whether we are working with decoding at end nodes or with distributed decoding, a coded message holds a maximum of one packet that is unknown to a given end node. Otherwise, the decoding is impossible at end nodes.

Since there are 2 end nodes in linear networks, the maximum number of packets in a coded message that are unknown at end nodes do not exceed 2. It is clear that with distributed decoding, the coded message cardinality is reduced to its minimum and that coded messages only hold packets that need to be delivered to end nodes. Distributed decoding can be seen as a neutralizer of redundant and unused copies of delivered packets created by NC.

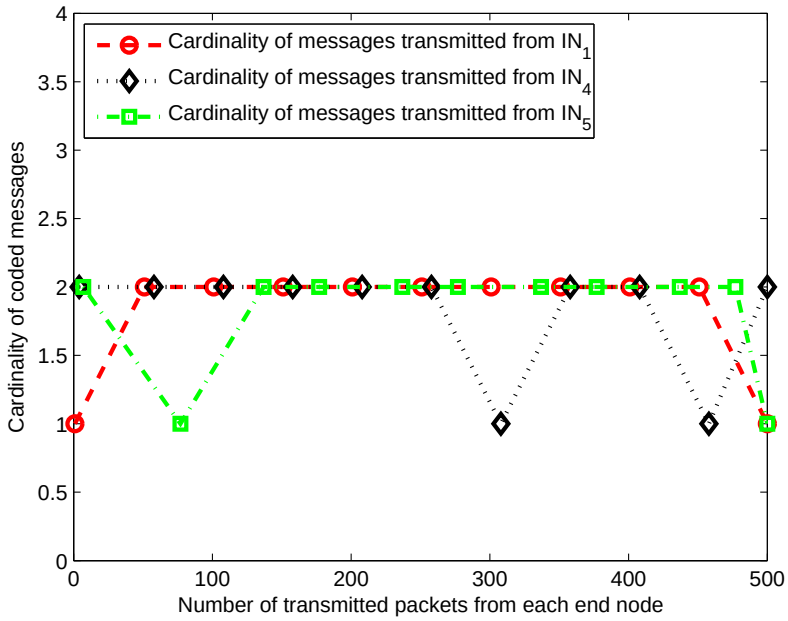


Figure 4.3: Cardinality of coded messages with distributed decoding (network of 8 nodes)

Figure 4.4 shows the number of neutralized packets at some specific nodes. There is a linear correlation between the number of iterations and the number of neutralized packets. The thin line on the figure represents equality between the number of iterations and the number of neutralized packets. Having all the plots below the thin is an indication that some transmissions do not hold redundant packets. This is due to the model used where QoS requirements might force the transmission of messages without being coded.

4.1.2 Buffering time counting

Buffering time is an important factor when it comes to resource management and has effects on the QoS. The buffering time of a packet in a node is computed, as the difference between the time the packet is received and buffered at the node and the last time the packet is used for decoding. The analytical counting of the buffering time with distributed decoding is given by:

Theorem 5. *The buffering time $B_t(i)$ at node i is given by:*

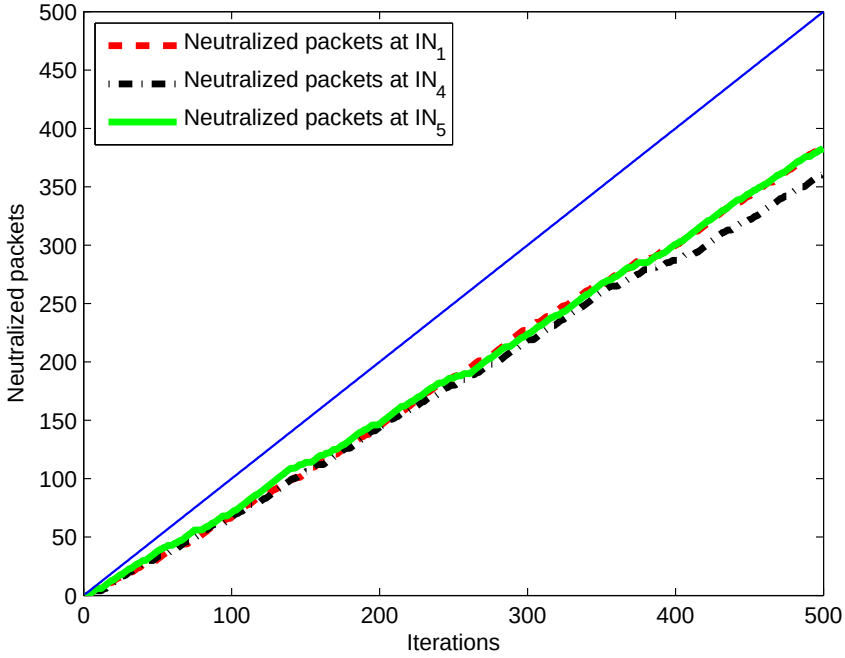


Figure 4.4: Number of neutralized packets (network of 8 nodes)

$$B_t(i) \quad = \quad q_s(i)t_m(i) + \max_j \ \{q_s(j)t_m(j)\} \ \ j \in N_i^+ \tag{4.1}$$

where $q_s(i)$ is the queue size at node i, $t_m(i)$ is the average time between two transmissions and N_i^+ is the set of nodes neighboring node i.

Proof. Consider the scenario shown in Figure 4.5 where a node IN_i receives a new packet j. The received packet is first buffered then coded with matching coded messages and placed in the queue of the node waiting to be broadcasted to the neighboring nodes. The worst case scenario occurs when the queue of the node is almost full. In this case, the time elapsed between receiving the packet and its transmission equals $q_s(IN_i) \times t_m(IN_i)$. Once transmitted, the same time takes place in each neighboring node before the packet is received back by node IN_i. After that, packet j is removed from coded messages in the neighboring nodes and will never reach node IN_i. The same scenario occurs at end nodes with both generated and received packets. ■

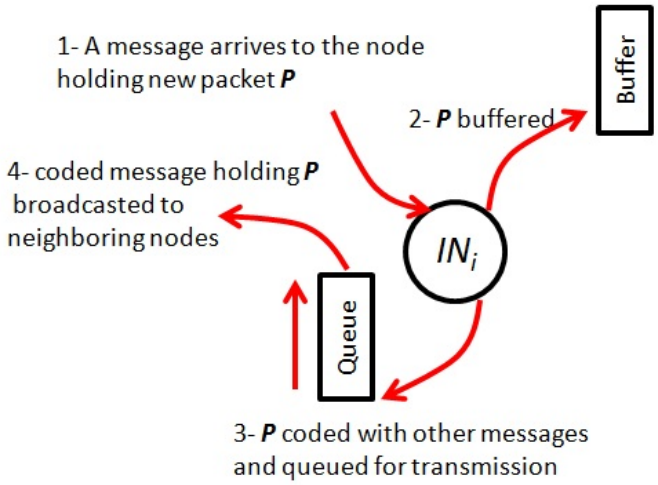


Figure 4.5: Buffering time diagram

4.1.3 Byte overhead transmission

Referring to Section 3.2.3, we now compare the byte overhead transmission for both centralized and distributed NC protocols. With distributed decoding, the maximum cardinality of a coded message is 2. Furthermore, the total number of transmissions by the $(N-2)$ intermediate nodes is $(N-2)\frac{P}{2}$ coded messages. thus, considering P_h as the size of the packet header, the number of extra bytes due to the coding activities is computed as $2 \times (N-2) \times \frac{P}{2} \times P_h$. Referring to 3.1 and with P_a the size in bytes of a packet, the number of bytes exchanged in the network for each protocol is summarized in Table 4.1.

Table 4.1: Lower bound on the number of bytes exchanged with each protocol

Protocol	Transmissions	Exchanged bytes	
Traditional store and forward	$(N-1)P$	$(N-1)PP_a$	(4.2)
NC-decoding at end nodes	$N\frac{P}{2}$	$N\frac{P}{2}P_a + P_h(N-2)\frac{P(P+2)}{8}$	(4.3)
NC-distributed decoding	$N\frac{P}{2}$	$N\frac{P}{2}P_a + P_h(N-2)P$	(4.4)

4.2 Performance evaluation

In this section the distributed decoding approach is evaluated from two points of view: the buffering time needed at each intermediate-node and the number of transmitted bytes engendered by the distributed decoding. Throughout this section and unless otherwise mentioned, we consider $P_a = 1500$ bytes (this size is normally used for Ethernet networks), $P_h = 15$ bytes and $P = 1000$ packets exchanged between the end nodes of a linear network of 8 nodes.

4.2.1 Buffering time simulation

Figure 4.6 reproduced from Chapter 3 and Figure 4.7 present, in percentage with respect to the total communication time, the highest and the average buffering time at each node of the network for decoding at end nodes and distributed decoding respectively. When decoding at end nodes, buffering is only needed at end nodes. However, the buffer size should be large enough to hold all exchanged packets since the largest buffering time is almost equal to the total transmission time needed to deliver all the packets. This is justified by the fact that the early generated packets might remain as part of the coded messages till the completion of the communication.

With distributed decoding where buffering is also needed at intermediate nodes, the buffering size is dramatically reduced. Indeed, the buffer time of a packet at any intermediate node does not exceed 2.7% of the total time needed to exchange 1000 packets. This buffering time corresponds to the time needed for a packet, newly transmitted from a node as part of a coded message, to be received back.

We also show in Figure 4.7 the buffering time computed by Theorem 5 giving a good estimation of the maximum buffering time needed at each node. The gain in buffer size is computed as the ratio between the total buffer size needed with decoding at end nodes and the total buffer size needed with distributed decoding. In our simulation and referring to Figures 4.6 and 4.7 this gain is of 93%. Finally, with decoding at end nodes, the buffering percentage with respect to the total communication time remains almost the same. This means that with decoding at end nodes, the buffer size increases with the number of exchanged packets while with distributed decoding, the buffer size at any node remains independent of the total number of exchanged packets.

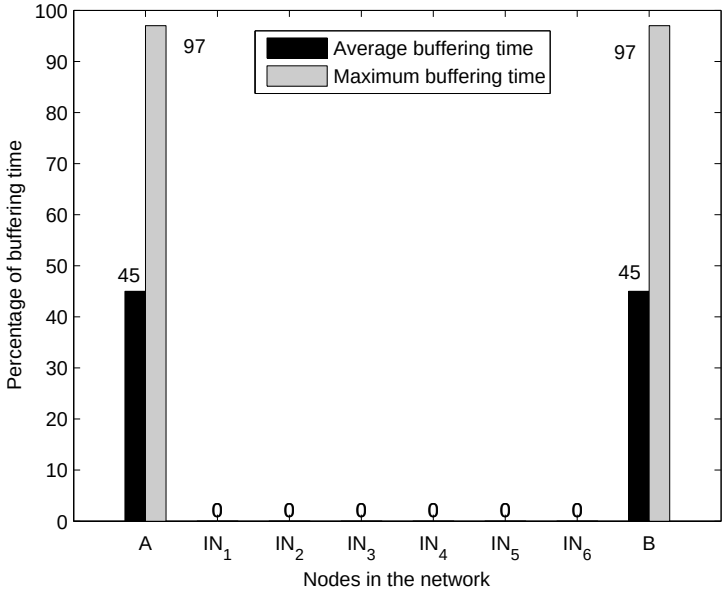


Figure 4.6: Decoding at end nodes: buffering time at each end node (1000 packets exchanged in network of 8 nodes)

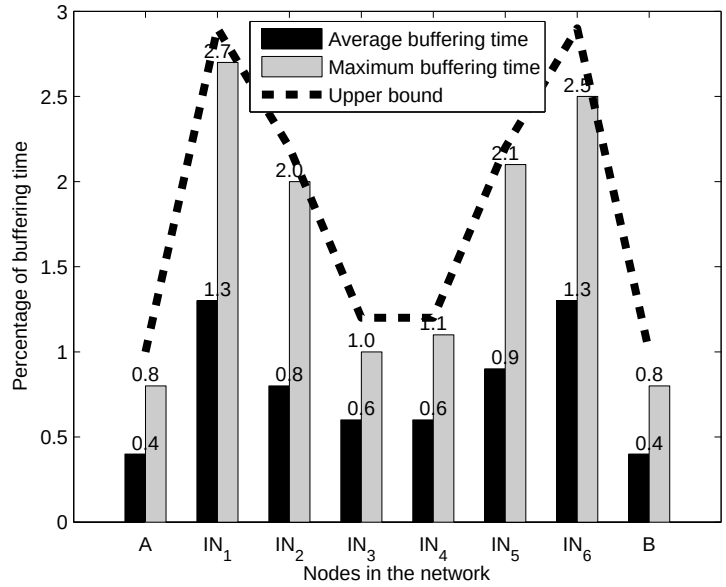


Figure 4.7: Distributed decoding: buffering time at each node (1000 packets exchanged in network of 8 nodes)

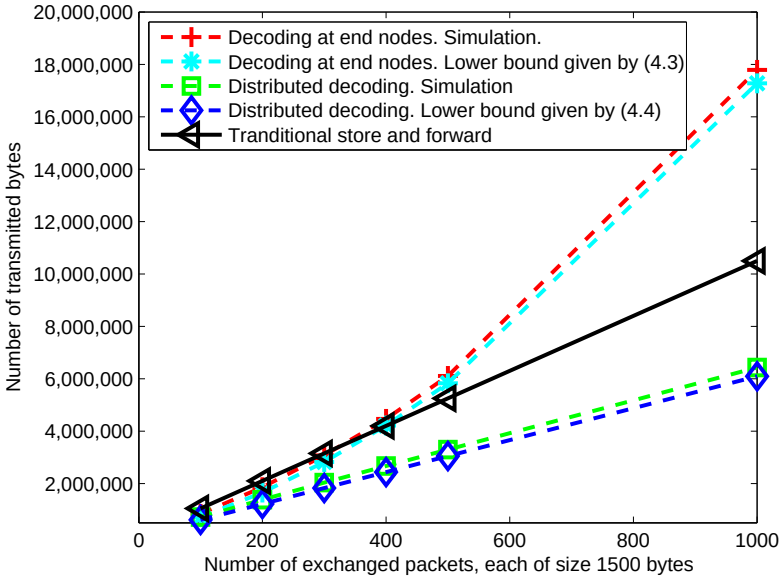


Figure 4.8: Number of bytes transmitted in the network for different protocols with $P_a = 1500$

4.2.2 Byte overhead transmission simulation

Figure 4.8 shows the number of transmitted bytes for each protocol and its corresponding lower bound that is given in Table 4.1. For traditional store and forward techniques, only one plot is shown since the real number of transmitted bytes coincides with (4.2). Figure 4.8 confirms our statement from Section 4.1.1 indicating that the growth in coded message cardinality increases the size of the header of the coded messages thus neutralizing at some point the advantages of NC. In fact, equalizing (4.2) and (4.3) gives the threshold on the number of transmitted packets for which NC with decoding at end nodes looses its advantages. However, distributed decoding reduces the byte overhead transmission to its minimum leading to an important saving in bandwidth and reenforcing the importance of NC. Indeed, the number of transmitted bytes needed to exchange 1000 packets between end nodes of the linear network is 17.3×10^6 when decoding at end nodes compared to 6.1×10^6 required for distributed decoding. An important reduction of 64% that favors our proposed distributed decoding protocol. Figure 4.8 confirms also the accuracy of our lower bound estimation on the number of transmitted bytes needed to exchange packets between end nodes of a linear network for

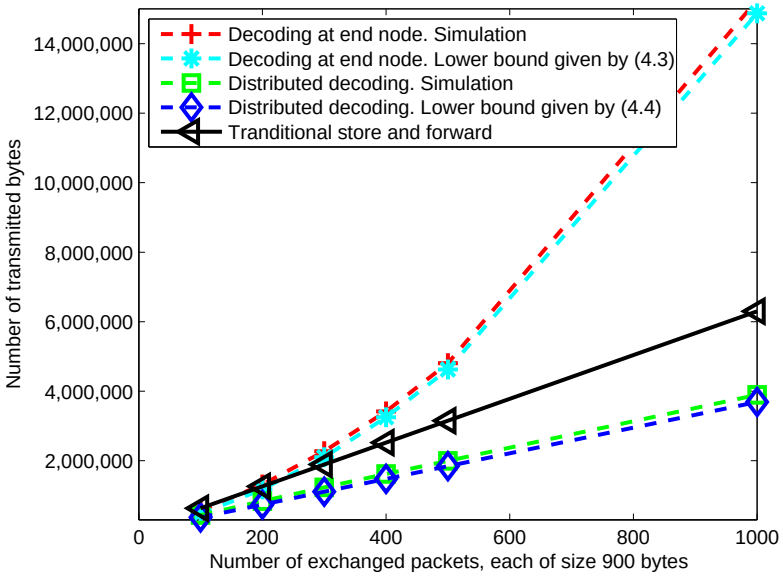


Figure 4.9: Number of bytes transmitted in the network for different protocols with $P_a = 900$

both NC with decoding at end nodes and NC with distributed decoding.

Furthermore, when reducing P_a to 900 bytes, Figure 4.9 shows the importance of using distributed decoding when NC is used on networks with small packet size. For the same number of exchanged packets, the reduction in the number of transmitted bytes reaches 74%. Without distributed decoding, NC has a reversed effect increasing the bandwidth usage instead of reducing it.

Figure 4.10 shows the number of bytes transmitted by each node in a network of 8 nodes and $P_a = 1500$. It is clear that distributed decoding reduces the byte overhead transmission at each node of the network.

4.3 Conclusion

In this chapter, we presented a novel solution for decoding messages in NC. This solution is based on a distributed approach that decodes packets at intermediate nodes. The decoding procedure removes redundant packets from coded messages at each node leading to a reduction in the number of exchanged bytes between end nodes. Simulations have shown a reduction of 64% in the number of transmitted bytes for our scheme

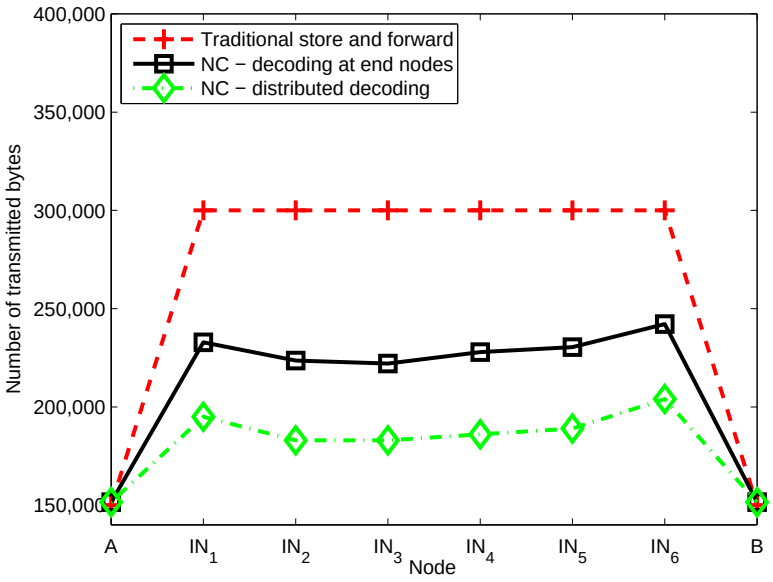


Figure 4.10: Number of bytes transmitted by each node to deliver 100 packets

compared to traditional decoding at end nodes when exchanging 1000 packets in a linear network of 8 nodes. We also proved that our approach reduces buffering and complexity overhead at end nodes. In fact, we showed by simulation that compared to decoding at end nodes, the buffer size is reduced by 93% when using distributed decoding to exchange 1000 packets. We also gave an accurate estimation on the number of transmitted bytes needed to exchange information between end nodes of a linear network for both NC protocols. At last, the study investigated in this chapter has been published in the IEEE wireless communications and networking international conference (WCNC) [C2].

CHAPTER

5 Impact of packet loss on the performance of NC

Without NC, lost packets have no impact on the delivery of other transmitted packets. With NC, the packet loss may affect the decoding of other transmitted packets thus affects the entire process of communication between nodes.

Since the publication of [3], researchers have concentrated on the use of NC for optimizing throughput in multicast networks [3, 11, 56] and for improving network reliability through random linear coding [16, 57]. However, only few published works have considered packet loss and packet loss recovery in the analysis of NC performance [58, 59].

In traditional communication protocols, reliability relies on retransmissions, and end to end reliability is ensured by the transport layer of the OSI layered communication stack. Transport layer uses ARQ, an error-control in data transmission that uses acknowledgments to achieve reliable data transmission over an unreliable service. Moreover, hop to hop reliability is ensured by PHY and MAC layers by the mean of HARQ, ARQ and forward error correction (FEC). In this chapter, ARQ refers to end to end ARQ of the transport layer.

Recently, hop by hop transport layer reliability protocols have been developed where intermediate nodes store and recover packets to deliver data more efficiently [60, 61]. With NC, reliability and throughput gain are achieved through linear NC [4, 11] where several packets are linearly combined together and broadcasted into the network. Several published works in the literature have dealt with the benefits of NC from the reliability point of view [57, 62].

In [57], authors have presented analytical and numerical results for the performance of end-to-end and link-by-link reliability mechanisms based on ARQ, FEC and NC in a tree topology. Using an access point topology with k receivers, they have demonstrated that the reliability gain of NC compared to ARQ is in $\Theta(\log k)$. They have also shown that allowing intermediate nodes to recover lost packets remains in $\Theta(\log k)$ while further minimizing the number of transmissions. However, it should be noted that buffering packets at intermediate nodes requires decoding coded messages which might be costly.

In this chapter, models presented in Section 3.1 are used to perform generating, coding, decoding and transmission activities on packets of the network. The impact of lost packets on buffering and complexity at the receiving nodes is studied and two new mechanisms are proposed to allow the recovery of the lost packets. Compared to traditional recovery protocols, our mechanisms provide a significant performance improvement in terms of number of transmissions required to recover from packet loss.

Due to lossy nature of wireless channel conditions, recent works have focused on loss recovery taking into consideration the overhead on the network due to retransmissions. In [63] authors have proposed *codecast* protocol suitable to low-loss, low-latency constraint applications such as video streaming. Their work is based on random NC [11] implementing localized loss recovery and path diversity with very low overhead. They demonstrated throughout simulations that *codecast* achieves a near-perfect packet delivery ratio while maintaining at the same time lower overhead than conventional multicast. They have concluded that routing and route selection have major impact on reliability and time delivery.

To reduce loss impact, routing selection mechanisms have been investigated. In [64] overlay routing networks have been proposed to monitor communication between nodes in order to improve routing of packets and provide loss reduction over traditional routing algorithms. Opportunistic routing [65–67] dynamically selects paths based on loss conditions. However, in [68] authors conducted a comparison of best-path routing and opportunistic routing and found that the benefit of opportunistic routing is much less than commonly believed. It should be well noted that these mechanisms introduces latency on the overall communication time and overhead on the routing nodes. The mechanisms we are proposing are network independent and require no probe from the network.

In the majority of cases, NC is considered as block coding where packets can only be decoded in batches [69]; in this context, a coded message, needs to wait for the arrival of other coded messages before being able to be decoded. While NC increases throughput and lowers congestion, block decoding may cause major delays and reduce QoS. Recent works have focused on QoS and proposed methods to reduce the average waiting time in the buffer of the intermediate nodes [54, 70]. Our work differs from existing works in the sense that we use NC for data exchange activity and not for multicast purposes. With our proposed ACNC protocol, no delay is engendered by the coding activity and no

packet is delayed waiting for a match. Moreover, instantaneous decoding at end nodes is guaranteed when packets arrive in order. Decoding remains guaranteed in real networks where messages arrive unordered.

Our contributions in this chapter are summarized as follows:

- For both decoding at end nodes and distributed decoding, the impact of packet loss on the decoding capability at the receivers and the decoding perturbation caused by each loss are investigated and analyzed.
- For decoding at end nodes, two new recovery mechanisms initiated by end nodes are proposed to handle packet loss with NC; i) immediate retransmission request (IRR) mechanism that uses information extracted from the undecodable messages to request the retransmission of lost packets. ii) recovery from packet loss mechanism with a basic covering set discovery (BCSD) that allows near optimal number of retransmissions to recover lost packets.
- For distributed decoding, a loss detection and recovery mechanism (LDRM) is presented. LDRM uses NC received messages as feedback for packet delivery and automatically adds potentially lost packets to the list of arrivals that needs to be transmitted.

5.1 Loss with centralized decoding

In a linear network of 4 nodes, Figure 5.1 shows a detailed trace of packets exchanged between end nodes A and B where node A generates packets P_{ai} $i = 1, 2, ...$ and node B generates packets P_{bi} $i = 1, 2, ...$. A maturity value $\mu = 3$ is set for the network and each packet is upper-indexed by its age. Recall from Chapter 3 that solid arrows indicate unicast activity to the next hop while dashed arrows indicate broadcast activity of newly coded messages. With the use of aging, redundant copies of packets quickly disappear from the network. For example, packet P_{a1} generated at iteration 1 is removed from all messages in the network at iteration 7.

In this context, we are mainly interested in the impact of losing a coded message, on the decoding capability at the receiving node. For example, if message $(P_{a1} \oplus P_{b2})$ between nodes IN_2 and B at iteration 3 is lost, node B successively receives coded messages that the node is unable to decode.

Figure 5.2 shows another trace conducted on a network with 5 nodes where the

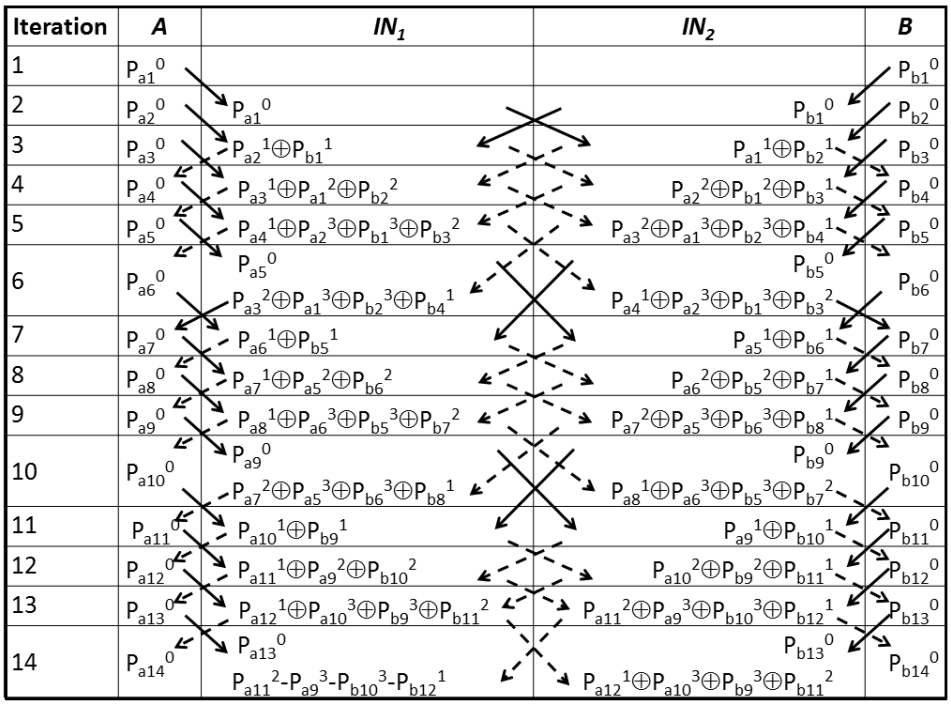


Figure 5.1: Delivery of packets in a network with no loss

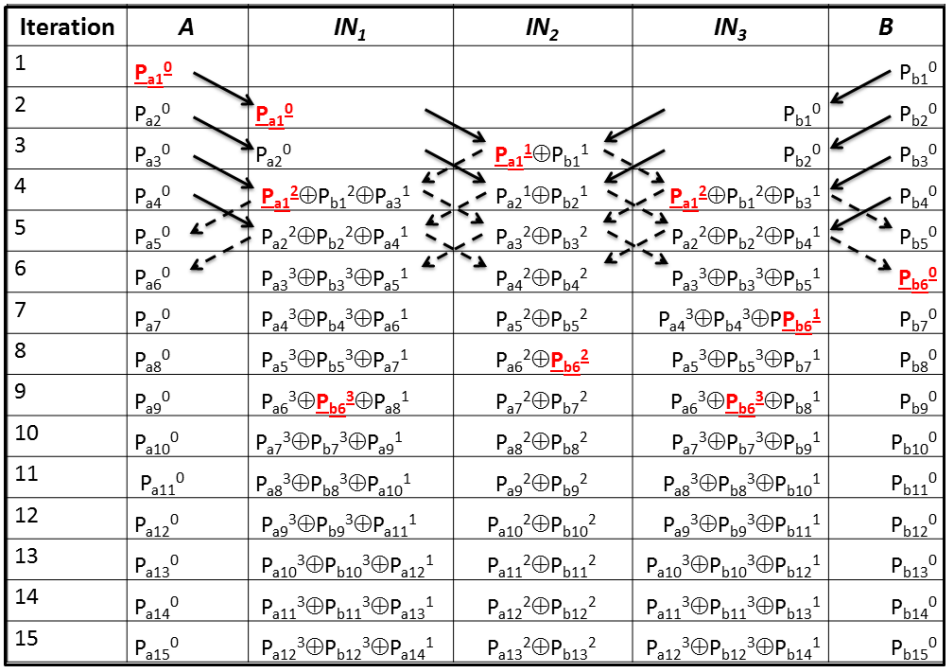


Figure 5.2: Delivery of packets in a network with 5 nodes

Iteration	A	IN_1	IN_2	IN_3	B
1	P_{a1}^{0}				P_{b1}^{0}
2	P_{a2}^{0}	P_{a1}^{0}		P_{b1}^{0}	P_{b2}^{0}
3	P_{a3}^{0}	P_{a2}^{0}	$P_{a1}^{1} \oplus P_{b1}^{1}$	P_{b2}^{0}	P_{b3}^{0}
4	P_{a4}^{0}	$P_{a1}^{2} \oplus P_{b1}^{2} \oplus P_{a3}^{1}$	$P_{a2}^{1} \oplus P_{b2}^{1}$	$P_{a1}^{2} \oplus P_{b1}^{2} \oplus P_{b3}^{1}$	P_{b4}^{0}
5	P_{a5}^{0}	$P_{a2}^{2} \oplus P_{b2}^{2} \oplus P_{a4}^{1}$	$P_{a3}^{2} \oplus P_{b3}^{2}$	$P_{a2}^{2} \oplus P_{b2}^{2} \oplus P_{b4}^{1}$	P_{b5}^{0}
6	P_{a6}^{0}	$P_{a3}^{3} \oplus P_{b3}^{3} \oplus P_{a5}^{1}$	$P_{a4}^{2} \oplus P_{b4}^{2}$	$P_{a3}^{3} \oplus P_{b3}^{3} \oplus P_{b5}^{1}$	P_{b6}^{0}
7	P_{a7}^{0}	$P_{a4}^{3} \oplus P_{b4}^{3} \oplus P_{a6}^{1}$	$P_{a5}^{2} \oplus P_{b5}^{2}$	$P_{a4}^{3} \oplus P_{b4}^{3} \oplus PP_{b6}^{1}$	P_{b7}^{0}
8	P_{a8}^{0}	$P_{a5}^{3} \oplus P_{b5}^{3} \oplus P_{a7}^{1}$	$P_{a6}^{2} \oplus P_{b6}^{2}$	P_{b7}^{0}	P_{b8}^{0}
9	P_{a9}^{0}	$P_{a6}^{3} \oplus P_{b6}^{3} \oplus P_{a8}^{1}$	P_{b7}^{0} $P_{a5}^{3} \oplus P_{b5}^{3} P_{a7}^{1}$	$P_{a6}^{3} \oplus P_{b6}^{3} \oplus P_{b8}^{1}$	P_{b9}^{0}
10	P_{a10}^{0}	$P_{b7}^{1} \oplus P_{a9}^{1}$	$P_{a8}^{2} \oplus P_{b8}^{2}$	$P_{a5}^{3} \oplus P_{b5}^{3} P_{a7}^{1}$ P_{b9}^{0}	P_{b10}^{0}
11	P_{a11}^{0}	$P_{a8}^{3} \oplus P_{b8}^{3} \oplus P_{a10}^{1}$	$P_{b7}^{2} \oplus P_{a9}^{2} \oplus P_{b9}^{1}$	$P_{a8}^{3} \oplus P_{b8}^{3} \oplus P_{b10}^{1}$	P_{b11}^{0}
12	P_{a12}^{0}	$P_{b7}^{3} \oplus P_{a9}^{3} \oplus P_{b9}^{2} \oplus P_{a11}^{1}$	$P_{a10}^{2} \oplus P_{b10}^{2}$	$P_{b7}^{3} \oplus P_{a9}^{3} \oplus P_{b9}^{2} \oplus P_{b11}^{1}$	P_{b12}^{0}
13	P_{a13}^{0}	$P_{a10}^{3} \oplus P_{b10}^{3} \oplus P_{a12}^{1}$	$P_{a11}^{2} \oplus P_{b11}^{2}$	$P_{a10}^{3} \oplus P_{b10}^{3} \oplus P_{b12}^{1}$	P_{b13}^{0}

Figure 5.3: Losing one packet in a network with 5 nodes

life cycle of two packets P_{a1} and P_{b6} is observed. The network is synchronized and transmissions occur only at the starting of each iteration. Note that in such case, the absorbing effect conserves a maximum cardinality of 3 for any generated coded message.

In Figure 5.3, a packet loss occurs at iteration 7 between IN_2 and IN_3 creating a discrepancy with Figure 5.2 that lasts till iteration 13. The perturbation caused by the packet loss increases both the number of transmissions as seen at iterations 9 and 10 and the cardinality of coded messages as seen at iteration 12.

5.1.1 Packet loss simulation

Algorithms presented in section 3.1 are used to run simulations on a linear network of 6 nodes, *i.e.* with 4 intermediate nodes. Statistics are recorded about packet loss impact on the decoding opportunity and buffering time at end nodes. In this simulation, only the hop $IN_3 \rightarrow IN_2$ is considered with loss and the packet loss is modeled as an independent and identical Bernoulli distributed process. The objectives of this simulation is twofold; first, understand the correlation between the packet loss and the number of undecodable packets at the receiver. Second, aid to establish a recovery process to minimize the number of retransmissions.

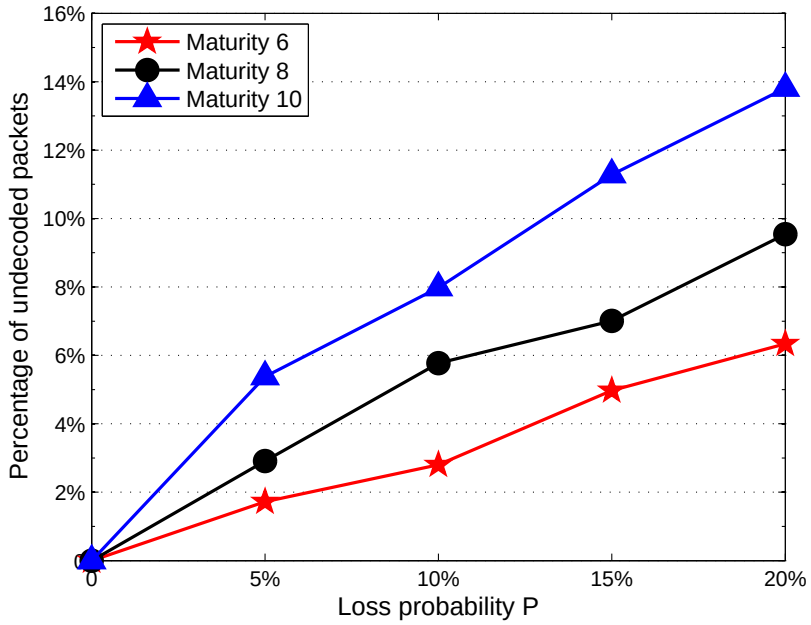


Figure 5.4: Percentage of undecodable packets with different loss probabilities and different maturities

5.1.2 Correlation between lost and undecodable messages

The percentage of undecoded packets is presented in figure Figure 5.4 for 10000 packets exchanged between nodes A and B is 10000 for different loss probabilities. The percentage of undecodable packets corresponds to the percentage of different packets that have been identified in the header of the received messages without being decoded. The simulation is repeated for 3 different maturity values. Figure 5.4 shows the impact of the aging concept on the number of undecodable messages. As shown in the figure, the number of undecodable packets at the receiving node increases with maturity. This is justified by the fact that when the maturity factor increases, coded packets live longer in the network and are further coded with other packets leading to an increase in the number of undecodable packets at the receiver. Moreover, the number of undecodable packets at the receiver is finite and the system resumes receiving decodable packets after each loss.

In Figure 5.5 the buffering of received packets at destinations is studied. Destination nodes need to buffer sent and received packets to be used in the decoding process.

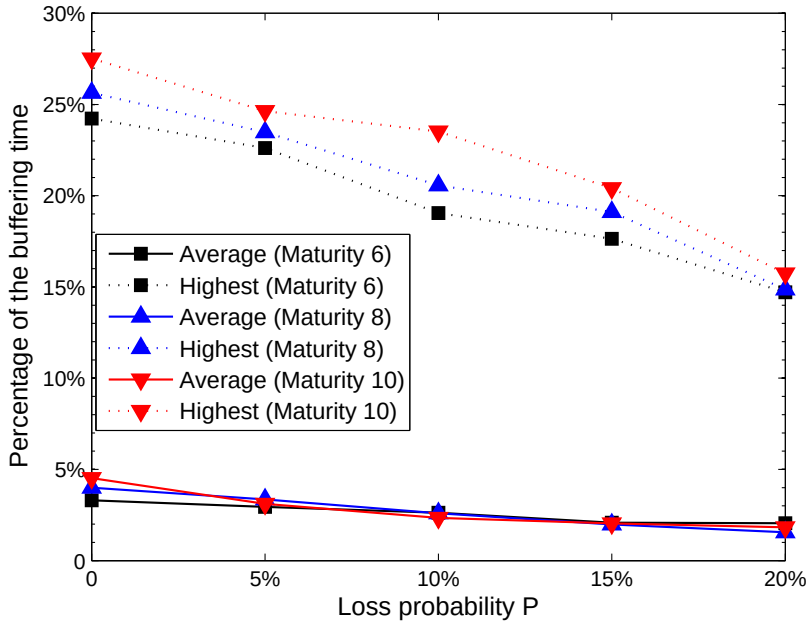


Figure 5.5: Buffering versus loss percentage

Buffering is costly since it requires significant memory space. Buffering time of a packet is computed as the laps of time between the moment the packet was received and the last time the packet was used for decoding other packets. The figure gives the percentage of the buffering time with respect to the total communication time versus the loss probability P, for 3 different maturity values, *i.e.* 6, 8 and 10. Furthermore, for each case, the highest values and the average values are represented. These results show that the buffering time decreases when the loss percentage increases. This is justified by the fact that less decoding occurs when more coded messages are lost. Moreover, the buffering also varies with maturity. When maturity increases, packets live longer in the network and the required maximum buffering time increases thus more buffering is needed.

5.1.3 Decoding opportunity

In this section we study the multiplicity of the lost packets in the network and its impact on the decoding process. By multiplicity, we mean the number of copies of the packet that exist in the network. Recall that, when a packet is coded with other packets, the resulting message is broadcasted into the network creating duplicates of

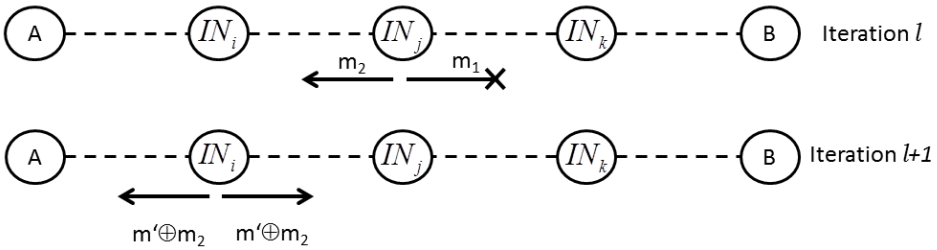


Figure 5.6: Lost packet discovery

these packets. When a packet is lost, other copies of that packet still exist in the network creating messages that cannot be decoded at the receiver.

To clarify how end nodes can receive undecodable messages, consider the example shown in Figure 5.6. In this example, a coded message m is created and broadcasted at iteration l by node IN_j (shown in the figure as m_1 and m_2 which are identical). Consider now that message m_1 is lost and m_2 reaches node IN_i. All packets in m_2 originated from A are flagged as exist at destination ("Exist at dest." flag set to true). Node IN_i uses message m_2 in further coding activity and broadcasts at iteration $l+1$ a new coded message $m^{'} \oplus m_2$. Node B cannot decode $m^{'} \oplus m_2$ since it holds more than one packet originated by A and unknown to B.

With a maturity μ, Theorem 3 presented in Chapter 3 sets a boundary on the dissemination of each packet in the network. Note that, each packet can be found in a maximum of 2^{μ} messages. Due to the linear nature of the network, each end node receives a maximum of $\frac{2^{\mu}}{2}$ messages containing the same packet. This upper bound on the multiplicity of a packet is used in the following theorem to estimate the number of undecodable messages when a packet is lost.

Theorem 6. *Given a network $G = (N, E)$ and a maturity scalar μ in $\mathbb{N}^+$. When a packet loss occurs in the direction of end node E, the estimated number of undecodable messages that can be received by E as a result of that loss is given by*

$$\frac{1}{\mu+1} \sum_{i=2}^{\mu} 2^{\mu-i} \tag{5.1}$$

Proof. If a packet in the direction of E is lost at age 0, *i.e.* before being coded, no other copies of that packet exist in the network. Consequently, no undecodable messages are received by E. On the other hand, if a packet is lost in the direction of E at maturity

age, the other copy of that packet is unicasted in the opposite direction. Consequently, no undecodable messages are received by E. Let P^a be a packet with age a, and N_{P^a} be the number of messages received by E that are created starting from P^a. According to Theorem 6, $N_{P^a} = \frac{2^{\mu-a}}{2}$. Referring to the coding graph presented in Chapter 3, we assume that in a network with maturity μ, there is an equal probability that a lost packet has an age between 0 and μ. The total number of undecodable messages that can be received by E is:

$$\frac{1}{\mu+1}\sum_{i=1}^{\mu-1} N_{P^i} = \frac{1}{\mu+1}\sum_{i=1}^{\mu-1}\frac{2^{\mu-i}}{2} = \frac{1}{\mu+1}\sum_{i=1}^{\mu-1} 2^{\mu-(i+1)}$$

By taking $i+1=j$, we can write

$$\frac{1}{\mu+1}\sum_{j=2}^{\mu} 2^{\mu-j}$$

■

As an example, a packet loss might lead to receiving 4 undecodable messages when maturity is set to 6, after which the system resumes receiving decodable messages.

Another important factor that helps reducing the number of received undecodable messages is related to the model presented in Section 3.1. Recall that the model allows packets to be transmitted without coding and never waits for a match. A simulation is performed to study the number of packets that are traveling through the network without being coded. Figure 5.7 shows the number of uncoded messages in the outgoing queue of each node of the network. End nodes generate packets while they receive the lowest percentage of undecodable messages. There is however, a significant percentage of uncoded packets floating between the nodes of the network. This is in fact due to the following factors:

- Aging: preventing packets from being further coded.
- Decision model: forcing packets to be transmitted without waiting for a match.

Analysis in this section shows an interesting characteristic of NC when it comes to lossy network. The intensive use of NC, represented by a large maturity value, keeps duplicates of a lost packet in the network and the number of undecodable packets increases. One of the advantages of aging is that it helps creating mature messages in the network thus preventing coupling with duplicates of lost packets.

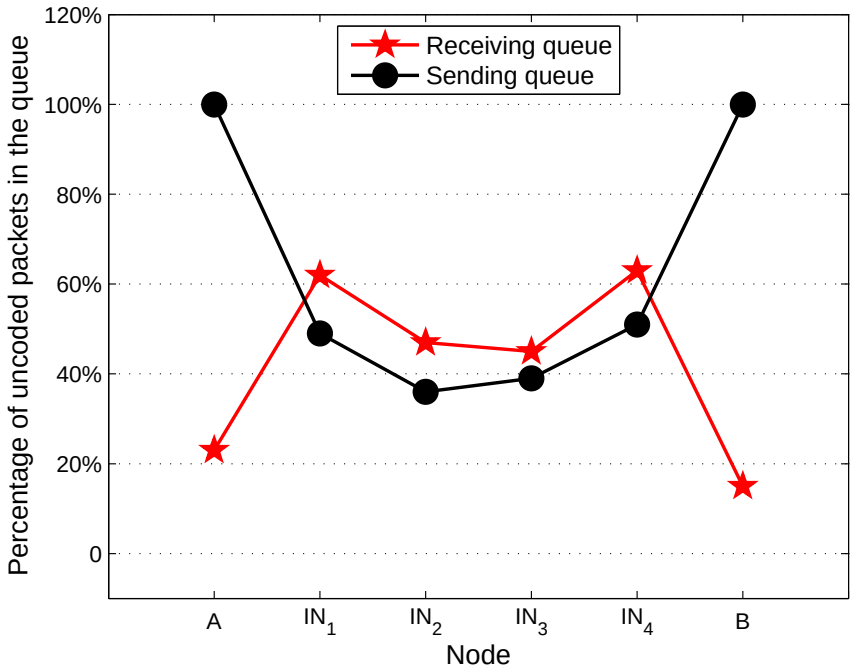


Figure 5.7: Percentage of uncoded messages at each node

5.2 Recovery from packet loss in centralized decoding

With NC, recovery from packet loss differs from traditional retransmission protocols since packets are interconnected together by the mean of coded messages. We have shown that losing one packet might lead to lose additional packets due to the nature of the coded messages and of the decoding process. For this reason, an efficient algorithm is needed in order to minimize the number of retransmissions.

To remedy from packet loss and its effect, two new mechanisms are detailed in this section. To further evaluate the performance of these mechanisms, a revised version of the LNC protocol is also presented first.

5.2.1 Revised LNC protocol

The revised LNC protocol uses block coding as follows; each sending node groups b packets together ($P_i \;\; i = 1 \; to \; b$) and sends b_Δ linearly independent combinations of the packets in each group. In this case the coding matrix M and the created combinations

$Y_j \;\; j = 1 \; to \; \Delta$ are illustrated as follows:

$$M = \begin{pmatrix} g_{11} & g_{12} & ..... & g_{1b} \\ g_{21} & g_{22} & ..... & g_{2b} \\ ... & ... & ..... & ... \\ g_{\Delta 1} & g_{\Delta 2} & ..... & g_{\Delta b} \end{pmatrix} \quad \text{with} \quad \underline{X} = \begin{pmatrix} P_1 \\ P_2 \\ ... \\ ... \\ P_b \end{pmatrix} \quad \text{and} \quad \underline{Y} = M\underline{X}$$

The cardinality b of each block and the number of combinations b_Δ are determined according to Theorem 6 and according to the loss percentage between the end nodes of the network given by:

$$P_n = 1 - (1 - P)^{N-1} \tag{5.2}$$

where N is the total number of nodes and P is the probability of loss in a link.

Once P_n is identified, the sending node needs to send $b_\Delta = \left\lceil \frac{b}{1-P_n} \right\rceil$ independent linear combinations of the packets of each block to guarantee the delivery of the packets in the block. In order to minimize the number of transmissions, the cardinality of the block should be selected to satisfy $\frac{b}{1-P_n} = \left\lceil \frac{b}{1-P_n} \right\rceil$.

Furthermore, each loss in the network leads according to Theorem 6, to have $U_n = \frac{1}{\mu+1} \sum_{i=2}^{\mu} 2^{\mu-i}$ undecodable messages that might reach the destination. With P_n as loss percentage in the network, $U_n \times P_n$ undecodable messages can reach the end node. To bypass the undecodability problem, b_Δ is augmented by $P_n \times b_\Delta \times U_n \times (1 - P_n)$.

5.2.2 Immediate retransmission request

The end node receiving an undecodable message is able to identify, by reading the header of the undecodable message, those packets having the "exist at dest." = true but are not in its buffered packets for decoding purpose. Without waiting for a retransmission event from the upper layer (*i.e.* transport layer) a retransmission request is initiated by the NC protocol for these packets.

Due to delays and unordered arrival of messages at the receiver, one undecodable message is not a sign of packet loss. The receiving node suspects that a loss occurs when undecodable messages remain undecodable for a specific period of time, denoted by t_{time} (threshold on time) or when receiving a certain number of consecutive undecodable

messages, denoted by t_{cup} (threshold on consecutive undecodable packets).

Algorithm 10 Immediate Retransmission Request

```
1: cup=0, i=1, rr=false, rp = ∅
2: while i ≤ number of UM do
3:     Add to rp the undelivered packets in UM(i)
4:     if UM(i) and UM(i-1) are consecutive then
5:         Increment cup
6:         if cup ≥ t_cup then
7:             rr = true
8:         end if
9:     else
10:        cup=0
11:    end if
12:    if Current time - Receiving time of UM(i) ≥ t_time then
13:        rr = true
14:    end if
15:    Increment i
16: end while
17: if rr = true then
18:    Request all needed packets in rp
19: end if
```

The Immediate Response Request (IRR) algorithm is designed to take into consideration the previously mentioned concerns. The IRR algorithm is shown in Algorithm 10 where UM denotes an undecodable message. The algorithm works as follows: initially, consecutive undecodable packets (cup) is set to 0, request retransmission (rr) is set to false and the list of requested packets (l_{rp}) is empty. The algorithm then iterates on each entry in the list of undecodable messages. Lines 4 to 11 are dedicated to identify consecutive undecodable messages and a request for retransmission is initiated if cup reaches a predefined threshold. Lines 12 to 14 investigate the buffering time of each undecodable message and request retransmission if a threshold on time is reached. The requests for packet retransmissions are sent in the header of newly generated packets or immediately if no new packets are available for transmission. Since no buffering of packets is performed at the intermediate nodes, the requests for retransmission is sent to the generator nodes. The algorithm is initiated each time the node is about to perform a decoding activity.

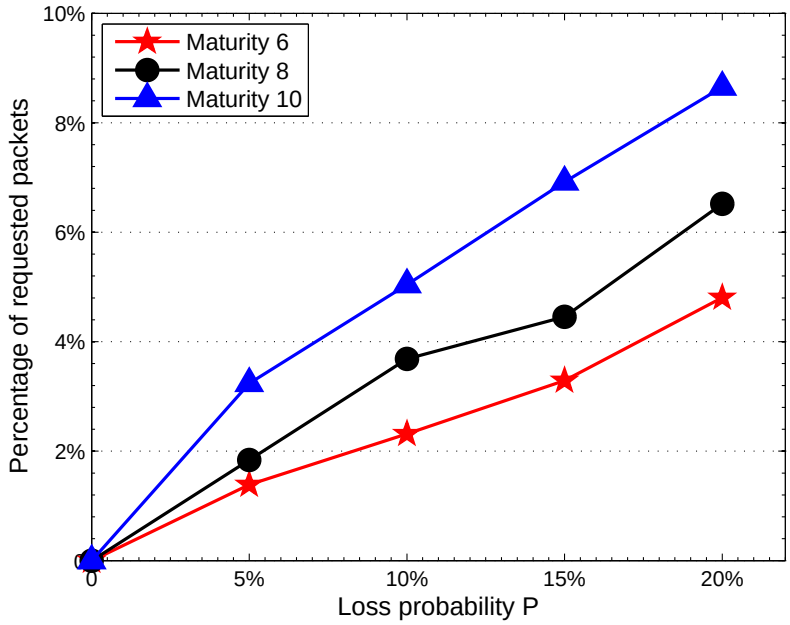


Figure 5.8: IRR: percentage of requested packets versus loss probability

IRR differs from the end to end ARQ of the transport layer. IRR is faster to both identify lost packets and to initiate a retransmission request. However, it should be noted that IRR does not replace ARQ. As shown in the proof of Theorem 6, a packet might be lost leaving no decoding consequences at the receiving nodes. Such lost can only be detected and requested by the transport layer using ARQ.

The simulation results of IRR are shown in Figure 5.8. The simulation ran with the 3 values of maturity and with the same conditions in Figure 5.4. Figure 5.8 shows that the number of requested packets increases with maturity values for the same packet loss percentage. Furthermore, the number of retransmissions compared to the number of received undecodable messages is presented in Table 5.2 for two loss percentages, where the undecodable and the requested columns are extracted from the results of Figures 5.4 and 5.8 respectively. Table 5.2 reveals a reduction in the percentage of requested packets when maturity increases. This is justified by the fact that, with larger maturity, a loss generates more undecodable messages and a retransmission helps also decoding more packets.

Table 5.1: Percentage of retransmission versus undecodable

	Loss probability P					
	10%			20%		
Maturity	undecodable	Requested	Percentage	undecodable	Requested	Percentage
6	280	205	73.2%	634	486	76.7%
8	577	371	64.3%	954	615	64.5%
10	798	503	63.0%	1381	865	62.6%

Table 5.2: Percentage of retransmission versus undecodable

5.2.3 Minimal retransmission request

Before presenting this new recovery process, we introduce some definitions and propose a mechanism to identify the packets to be requested from the sender in order to recover all lost packets with minimal retransmissions.

Closures and covering sets

Let S be a set of packets and C be a set of coded messages.

Definition 20. *When XOR operation is used to code packets together, the* ***simple closure*** *of S, denoted by S_C^+, is the set of packets that can be decoded from C using packets in S.*

To better understand the simple closure definition, we illustrate the following example.

Let $S = \{p_1, p_2, p_3\}$ and $C = \{p_1p_4, p_1p_3p_5, p_4p_5p_7, p_5p_6p_7, p_6p_8p_9, p_7p_8p_9\}$
To build the simple closure of S, we proceed as follows:

$$\begin{aligned}
S_C^+ &= \{p_1, p_2, p_3\} \\
S_C^+ &= \{p_1, p_2, p_3, p_4\} && \text{using} \quad p_1p_4 \\
S_C^+ &= \{p_1, p_2, p_3, p_4, p_5\} && \text{using} \quad p_1p_3p_5 \\
S_C^+ &= \{p_1, p_2, p_3, p_4, p_5, p_7\} && \text{using} \quad p_4p_5p_7 \\
S_C^+ &= \{p_1, p_2, p_3, p_4, p_5, p_7, p_6\} && \text{using} \quad p_5p_6p_7
\end{aligned}$$

Definition 21. *When linear combination is used to code packets together, the* ***complex closure*** *of S, denoted by S_C^{++}, is the set of packets that can be decoded, by the mean of Gaussian Elimination, from C using packets in S.*

Going back to the previous example, the complex closure of S, is denoted by:

$$S_C^{++} = S_C^{+} \cup \{p_8, p_9\}$$

where p_8 and p_9 can be obtained by solving the system:

$$\begin{cases} \alpha_1 p_8 + \alpha_2 p_9 &= R_1 \\ \beta_1 p_8 + \beta_2 p_9 &= R_2 \end{cases}$$

Where R_1 and R_2 are the received coded packets and $(\alpha_1, \alpha_2)^t$ and $(\beta_1, \beta_2)^t$ are the corresponding coding vectors normally included in the header of the received coded packets.

Definition 22. *S is a **covering set** of C if the closure of S includes all packets that are part of the coded messages in C.*

Going back again to the previous example, we see that S_C^{++} is a covering set of C but S_C^{+} is not.

Definition 23. *A **basic covering set** of C is a covering set of C with minimal cardinality (having the least number of packets).*

Basic covering set discovery

The recovery process starts when end node receives a message that cannot be decoded, *i.e.* it has more than one unknown packet. At that time, the end node expects to receive additional undecodable packets. Each received undecodable message is XOR-ed with all known packets and the resulting chunk of undecodable packets XOR-ed together is added to an originally empty set C.

The gathering period of undecodable chunks of messages is controlled by a timer that starts immediately after the reception of the first undecodable message. As stated before, undecodability might be caused by unordered arrival of messages and the problem might be solved without intervention when more messages arrive to the end node. The duration of the timer is an application specific parameter based on the communication medium and the type of application. It can vary from milliseconds in the case of instant messaging where messages need to be delivered in order to the upper layer, to seconds in the case of file exchange where only the complete file needs to be ordered. When the timer times out, the recovery mechanism is lunched. The recovery mechanism consists of finding a basic covering set and packets in the covering set are requested from the sender. The Basic Covering Set Discovery (BCSD) algorithm is presented in Algorithm

Algorithm 11 Basic covering set discovery (BCSD)

```
1: Build the incident matrix M
2: B an empty set
3: repeat
4:     i = 1
5:     maxj = 0
6:     max1 = 0
7:     for j = 1 to NbColumns do
8:         if sum of 1's in column j > max1 then
9:             maxj = j
10:            max1 = sum of 1's in column j
11:        end if
12:    end for
13:    add maxj to B
14:    Remove column maxj
15:    repeat
16:        if there is a row with one entry then
17:            Remove the row and column of that entry
18:        end if
19:    until No more columns can be removed
20: until Matrix is empty
21: Request packets in B from the sender
```

11. The algorithm follows a greedy mechanism that tries, within a reasonable amount of time to identify a basic covering set for all chunks of messages in C. The algorithm starts by building the incident matrix where columns identify the packets in C and rows represent undecodable chunks of messages sorted by receiving time. For the set $C = \{p_1p_4, p_1p_3p_5, p_4p_5p_7, p_5p_6p_7, p_6p_8p_9, p_7p_8p_9\}$ the corresponding incident matrix M is:

$$M = \begin{array}{c} \\ p_1p_4 \\ p_1p_3p_5 \\ p_4p_5p_7 \\ p_5p_6p_7 \\ p_6p_8p_9 \\ p_7p_8p_9 \end{array} \begin{array}{c} \begin{array}{cccccccc} p_1 & p_3 & p_4 & p_5 & p_6 & p_7 & p_8 & p_9 \end{array} \\ \begin{pmatrix} 1 & & 1 & & & & & \\ 1 & 1 & & 1 & & & & \\ & & 1 & 1 & & 1 & & \\ & & & 1 & 1 & 1 & & \\ & & & & 1 & & 1 & 1 \\ & & & & & 1 & 1 & 1 \end{pmatrix} \end{array}$$

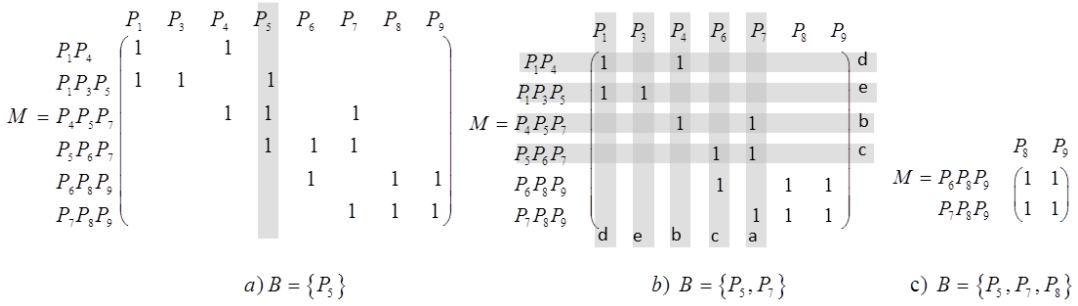


Figure 5.9: BCSD example

The algorithm then iterates on the incident matrix until the covering set is built. At each iteration, the column with the highest number if 1's is selected (*i.e.* highest occurrence in the undecodable messages) and the corresponding packet is added to the covering set B. If more than one row with the same highest number of 1's are present, then any one of these rows can be selected. Rows with one entry are then removed together with the column of the entry since the corresponding packet can be decoded. The algorithm stops when the incident matrix becomes empty.

Steps applying BCSD algorithm to the incident matrix M are shown in Figure 5.9 and the basic covering set found is $\{p_5, p_7, p_8\}$. First the algorithm selects P_5 to be in the covering set since it corresponds to the column with the highest number of 1's. The associated column is removed and the resulting matrix is shown in Figure 5.9b. The second iteration of the algorithm selects P_7. However, removing its associated column leaves rows b and c with exactly one packet each so both columns b and c and rows b and c are removed creating an additional decoding opportunity at rows d and e. Figure 5.9c shows the resulting matrix at the end of iteration 2. Finally, packet P_8 (or P_9) is selected at iteration 3 to complete the covering set.

5.2.4 Simulation results

In this section, both IRR and BCSD algorithms are implemented and tested. Simulations run on a linear network of 6 nodes when a loss probability is applied on each link. Results are compared to both traditional store and forward with ARQ protocols and to the revised LNC protocol. Note that the revised LNC uses the same decision models presented in Section 3.1. Along the simulations, 10000 packets are exchanged

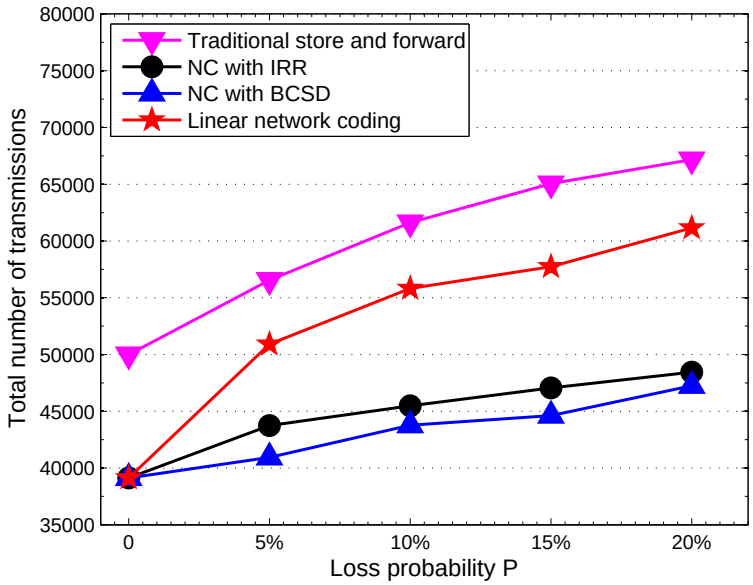


Figure 5.10: Total number of transmissions required to deliver 10000 packets

between both end nodes of the network. Algorithms are compared according to the total number of transmissions needed to deliver all the exchanged packets between the end nodes. In the SF protocol, the transport layer at the end nodes automatically claims unacknowledged packets. The total number of transmissions to reliably deliver all the packets is recorded. In a reliable network of 6 nodes, 5 transmissions are needed to deliver each packet from one end node to another. The performance of IRR and BCSD are also evaluated and compared to LNC and SF.

Results are presented in Figure 5.10. Traditional store and forward algorithm starts with a total of 50000 transmissions on a reliable network and ends with an increase of 34% when the loss probability reaches 20%.

Moreover, on a reliable network, revised LNC, IRR and BCSD start with almost the same number of required transmissions to deliver all the packets. When the loss probability increases, the number of transmissions with the revised LNC increases about 56% when the loss probability reaches 20%. Both IRR and BCSD show an important saving in the number of transmissions when compared to the revised LNC. With a loss probability of 20%, BCSD presents a saving of 22% in the number of transmissions over

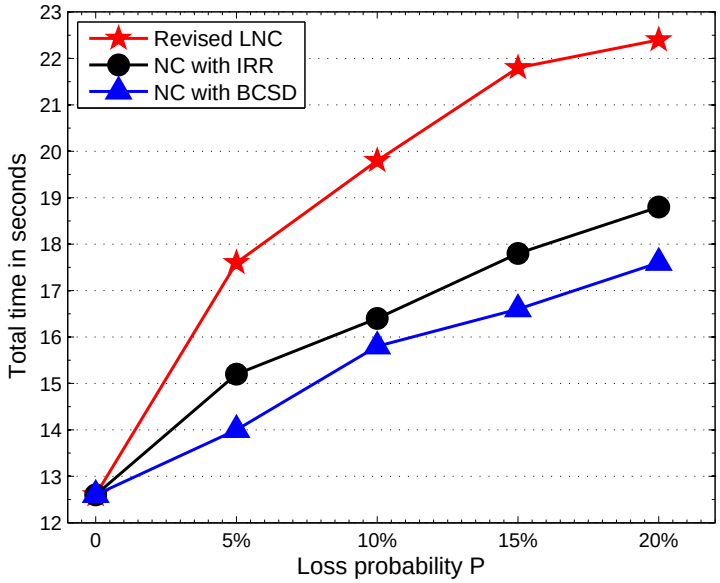


Figure 5.11: Total delivery time of the 10000 packets exchanged

the revised LNC. IRR slightly increases the number of transmissions compared to BCSD.

Figures 5.11 and 5.12 show respectively the total delivery time and the average delivery time for the three competing mechanisms. The total delivery time is computed as the time elapsed between the transmission of the first packet and the reception of the last one while the delivery time of a packet is computed as the time between the first transmission of the packet and its delivery. In these simulations, each transmission from one node to an adjacent one requires 1 millisecond of time. As shown in Figure 5.12, the revised LNC overcomes the proposed mechanisms when it comes to average delivery time. This is justified by the fact that no retransmission is requested. With the revised LNC, the slow increase in the average delivery time is due to the time needed at the receiver to gather enough messages of each block in order to decode the packets of the block. As for our proposed mechanisms, IRR bypasses BCSD when it comes to average delivery time, since IRR reacts immediately when a loss is detected while BCSD waits longer to gather undecodable messages and reduce the size of the covering set, *i.e.* the number of retransmissions.

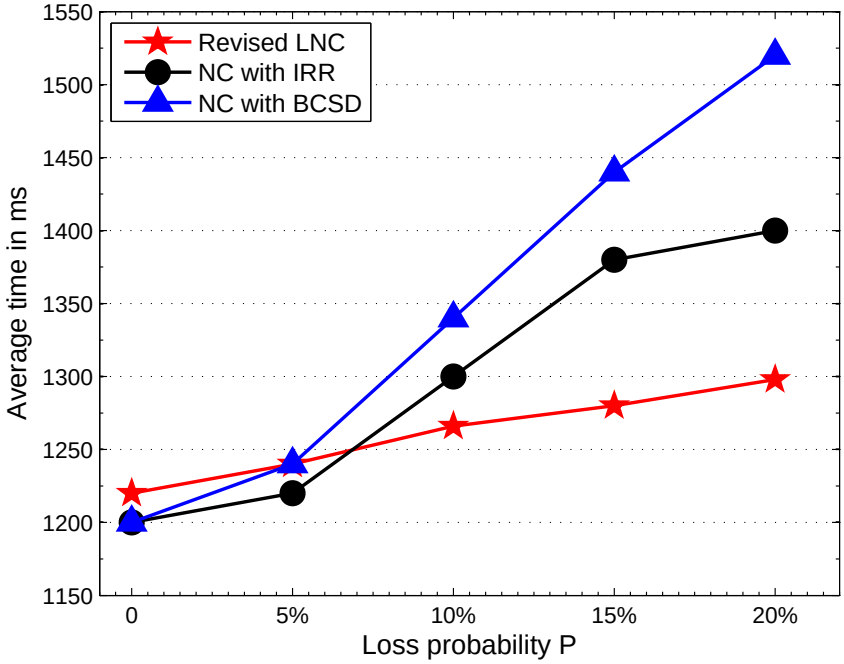


Figure 5.12: Average delivery time.

5.3 Loss with distributed decoding

In this section, a new coding mechanism is provided to automatically detect and recover lost packets in wireless lossy networks that employs distributed decoding. Contrary to the majority of ARQ algorithms relying on acknowledgment, our mechanism extracts feedback on packet delivery by analyzing received network coded messages and automatically reacts when packet loss is sensed.

In WSN and WMN, channels are subject to interference, fading, noise and limited bandwidth. ARQ is a conventional method for error recovery applied in a hop by hop manner at the link layer of the OSI model or in an end-to-end manner at the transport layer. ARQ requires the sender to be aware of the packet loss using negative acknowledgment (NACK) or timeout before requesting or initiating retransmission of the lost packet. This process is time consuming and not suitable for real time communication. Fault-tolerant techniques in WSN and WMN initially focus on routing mechanisms to improve reliability in an efficient way. We now propose a mechanism able to deal with loss in a distributed fashion.

5.3.1 Reliable data transfer

With distributed decoding, intermediate nodes buffer packets to be used for decoding. Our novel loss recovery protocol is based on the idea of taking advantage of this buffering mechanism to retransmit packets when a loss is detected. With the proposed protocol, the loss recovery process is decentralized and no acknowledgment or packet retransmission request is needed.

Definition 24. *In networking, Promiscuous mode is a mode for a wired or wireless network interface controller that causes the controller to pass all traffic it receives to the central processing unit rather than passing only the frames that the controller is intended to receive.*

When node i transmits, to neighboring nodes, a message containing a new packet P_r that is newly arrived to i, it expects to receive back exactly one coded message, for an example M_{P_r} containing packet P_r. If no coding opportunity is found for P_r at a neighboring node, P_r is unicasted to its destination and node i, initially set in promiscuous mode overhears the transmission of P_r. If P_r is not received or overheard in a message within a specific amount of time, node i initiates a loss recovery process by adding packet P_r to the receiving queue for a possible coding and transmission activity.

The proposed mechanism requires end nodes to acknowledge newly received packets and hence distributed decoding protocol is modified to enable this acknowledgment without additional transmissions. End nodes code with each new packet ready to be transmitted, all previously received packets since the last transmission. The resulting coded message is unicasted to the next hop of the newly generated packet. If there exist no packet to transmit, received packets from the same source since the last transmission are coded and sent to that source.

There are two possible reasons for node i not to receive or overhear message M_{P_r}:

1. M_{P_r} was lost during its transmission to neighboring node.
2. M_{P_r} was lost when transmitted to node i.

Before going further with our loss detection and recovery mechanism (LDRM), some definitions are needed to clarify some of its aspects.

Definition 25. *The round trip $RT_{ij}(P_r)$ of a packet P_r between nodes i and j is defined as the time elapsed between the transmission of P_r as a single packet or within a coded message and the time P_r is received back from node j coded within a message or overheard. Nodes i and j can be intermediate or end nodes.*

Proposition 1. *Given a transmission rate R, a propagation speed S, the distance D between nodes i and j, and a coded message length of L bits, the round trip $RT_{ij}(P_r)$ is given by:*

$$RT_{ij}(P_r) \quad = \quad 2 \times \left(\frac{L}{R} + \frac{D}{S} \right) + B_{t_j} \tag{5.3}$$

where B_{t_j} is the average buffering time at node j.

Proof. By definition, the round trip of a packet equals to the sum of its transmission time from i $\left(\frac{L}{R}\right)$, its propagation time to j $\left(\frac{D}{S}\right)$ its buffering time at j and the trip back. ■

Definition 26. *The set i^+ is defined as the set of all neighboring nodes to i.*

Definition 27. *The round trip $RT_i^+(P_r)$ of a packet P_r between node i and all nodes in i^+ is defined as the time elapsed between the transmission of P_r as a single packet or within a coded message and the time P_r is received back or overheard from every node in i^+.*

Proposition 2. *The rount trip $RT_i^+(P_r)$ is given by*

$$RT_i^+(P_r) \quad = \quad \max_j \; RT_{ij}(P_r) \; j \in i^+ \tag{5.4}$$

Algorithm 12 describes our LDRM and works as follows: when a message is received by node i from a neighboring node k, packets in the message are scanned and compared to buffered packets in B (lines 1, 2 and 3). When a match is found, say for packet P_m, buffered packet P_m is removed from the message by XORing P_m it with the message. The header of the message is also updated (lines 4 and 5). As part of the loss detection and recovery process, packet P_m in B is flagged as being received by node k. On the other hand, being already flagged as received by k is an indication that node k did not receive from i a message containing P_m and a retransmission is required. In this case, the P_m is included in the next outgoing message to k as an acknowledgment (lines 6, 7, 8 and 9). After removing all matching packets from the received message, the cardinality of the message is checked. If the cardinality is set to 1, the remaining packet is new to node i and it is added to the buffer B and its buffering timer starts as part of the loss detection process (lines 14, 15 and 16). If a buffered packet times out in B and it is not flagged as received by all neighboring nodes, the packet is broadcasted to neighboring nodes and the timer of that packet is reset (lines 18, 19, 20 and 21).

Algorithm 12 LDRM: when receiving packet from k

```
for each packet P_i in the received message do
    for each packet P_j in the buffer B do
        if P_i = P_j then
            message = message ⊕ P_j
            Remove from the message the header of P_i
            if P_j not flagged as received by k then
                flag P_j as received by k
            else
                include P_j in the list of packets to k
            end if
        end if
    end for
end for
if cardinality of the message is 1 then
    add the single packet to B
    start buffering timer for the newly added packet
end if
if a packet in B times out then
    if Not received by all neighboring nodes then
        add the timed out packet to the outgoing queue
        reset the timer
    end if
end if
```

5.3.2 Simulation results

The LDRM algorithm is validated by running simulations on linear network of 6 nodes. Simulations are completed using a self-developed JAVA-based application that simulates the network layer at each node of the network and the transport layer at end nodes for traditional end-to-end ARQ technique. The packet loss is modeled as an independent and identical Bernoulli distributed process on all links.

LDRM is compared with a modified version of the distributed decoding protocol presented in Section 4.1 that uses end-to-end ARQ to recover from packet loss. The modified version, named DNC-ARQ works as follows: an acknowledgment (ACK) is sent by end nodes for each received and decoded packet. ACKs are grouped and included in the transmitted packets, or sent individually if no packet has to be transmitted.

Figure 5.13 shows the number of transmissions required to exchange 10000 packets

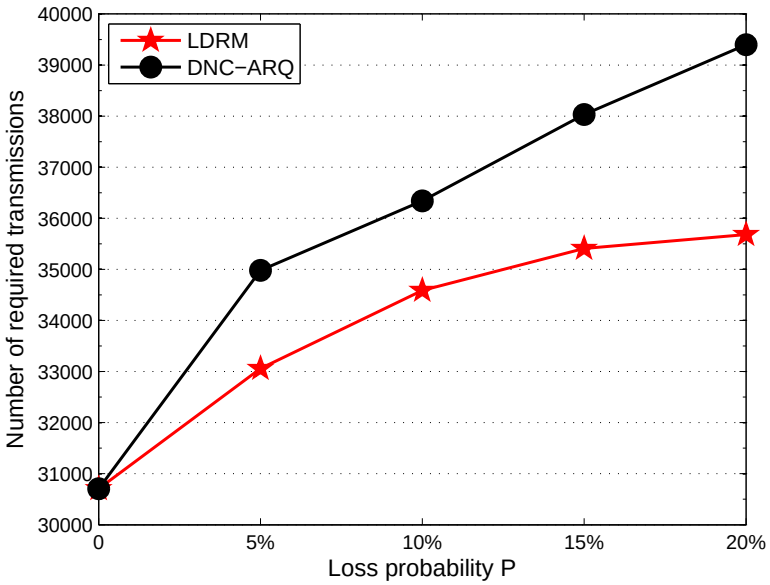


Figure 5.13: Total number of transmissions needed to deliver 10000 packets

(5000 packets generated by each end node) between the two end nodes of the network. The simulation is run for different loss percentages and the number of transmissions needed is recorded for both LDRM and DNC-ARQ mechanisms. It can be observed that the proposed scheme outperforms the traditional ARQ process in terms of number of retransmissions when the loss probability on each link increases. For a network with 20% loss probability, the number of saved transmissions with LDRM exceeds 35000. This is due to 1) ARQ is an end-to-end mechanism while LDRM is a hop-by-hop mechanism. Thus one packet loss with ARQ requires retransmitted packet to travel over 5 links while a packet loss with LDRM requires a packet to travel only to next hop.

Figure 5.14 shows the time needed to deliver all 10000 packets for both algorithms. In these simulations, each transmission from one node to an adjacent one requires 1 millisecond of time. The total delivery time is the time elapsed between the transmission of the first packet and the reception of the last one. As it can be seen on Figure 5.14, both algorithms have similar performances though LDRM still slightly outperforms DNC-ARQ.

Figure 5.15 shows the average delivery time of a packet between end nodes. This

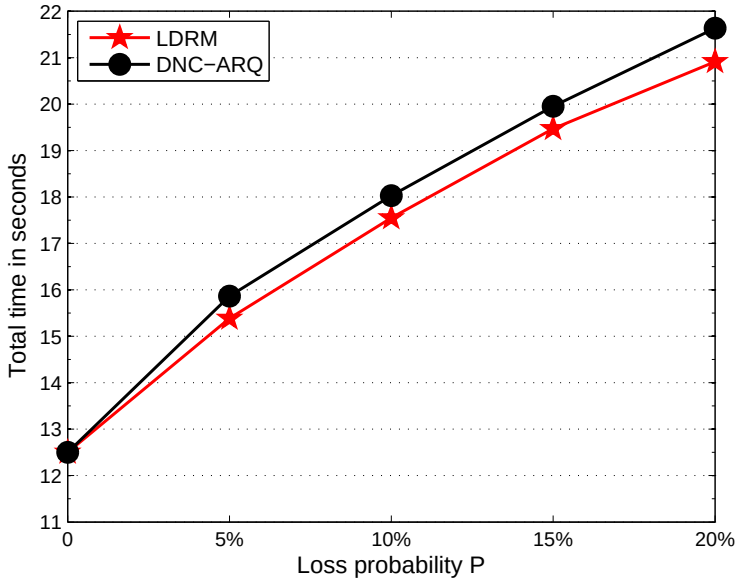


Figure 5.14: Total time needed to deliver 10000 packets

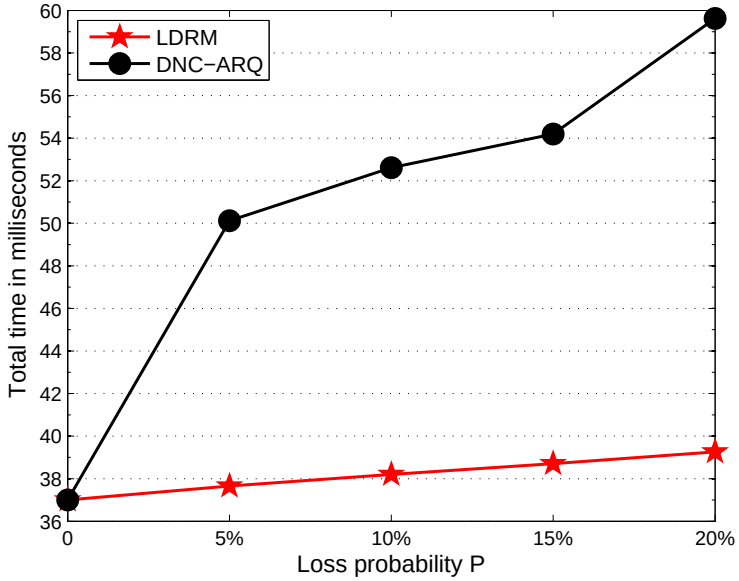


Figure 5.15: Average time needed to reliably deliver a packet between end nodes

is the time for a packet, in average, to travel from an end node to another. The weak performance of DNC-ARQ is clear from this figure and justified by the fact that for each loss, an end node needs at least 5 transmissions time to report a loss and another 5 transmission times to receive lost packet. Moreover, the hop-by-hop loss detection is from far, faster than an end-to-end loss detection.

5.4 Conclusion

This chapter dealt with the impact of packet loss in NC, where the models presented in Section 3.1 have been used to study packet loss effect on the decoding process at end nodes. Simulations have shown that with the proposed model and with the use of aging, any perturbation caused by losing packets is finite in time and end nodes receive a limited number of undecodable packets. The number of undecodable packets at end nodes has been analytically and experimentally studied under the effect of aging and NC activities where two new mechanisms have been proposed to improve NC performance in lossy networks. The first mechanism, called IRR, immediately requests lost packets when a packet loss occurs and the second mechanism, called BCSD, requests missing packets in a reduced number of transmissions. Simulation results have shown that in typical cases, both algorithms help reducing by 22% the number of transmissions comparing to the revised LNC protocol. Finally it should be noted that IRR and the revised LNC are more suitable to some applications like video conferencing where the average delivery time is needed to be reduced while BCSD is more convenient to file exchange where the total delivery time needs to be optimized.

We have also studied in this chapter the loss problem with distributed decoding and proposed LDRM algorithm for loss recovery. Contrary to the majority of ARQ mechanisms, LDRM uses NC received messages as feedback for packet delivery and automatically adds potentially lost packets to the list of arrivals that needs to be transmitted. Simulation has shown that LDRM not only reduces the number of required transmissions comparing to traditional end-to-end ARQ but also improves the delivery time of packets thus enhancing the QoS. At last, the impact of loss with distributed decoding investigated in this chapter has been accepted in 23rd international conference on Software, Telecommunications and Computer Networks (SoftCOM 2015) [C1] while the impact of loss with decoding at end nodes has been part of the journal paper submitted

to EURASIP [A1].

Conclusion and Perspectives

Conclusion

In this thesis, we have proposed a new network coding protocol based on address correlation (ACNC) of packets exchanged between end nodes. This protocol is based on two decoding methods, the centralized decoding taking place at the end nodes and the distributed decoding where all intermediate nodes participate in the decoding process. We have started this thesis with a general overview on the NC theory, focusing mainly on the advantages of this technology. It is well noted that the continuous increase on bandwidth demand makes such technology a requirement for future communication. Then, we have listed the wide application areas of NC, which can be divided into bandwidth saving and reliability .

Since this thesis focuses on both bandwidth saving and reliability, we have introduced in Chapter 2, the ACNC protocol as the starting point of our studies. ACNC is a complete NC based protocol that guarantees a certain level of QoS and 100% decodability at the receiving nodes. Both centralized and distributed decoding are based on ACNC.

In the next part of the thesis, namely in Chapter 3, we have presented the centralized decoding protocol where decoding occurs at the receiving nodes only. With this approach, intermediate nodes are only responsible for coding and forwarding coded messages to destinations. This approach is studied from different perspectives:

First, we have presented two decision models to mange intermediate and end node activities respectively. For intermediate nodes, the decision model is deterministic and tries to balance the occupation of the receiving and the outgoing queues. By doing so, the risk of packet drop due to receiving queue being full is reduced. For end nodes, a probabilistic model is used that better simulates the packet generation and delivery to

the upper layer of the OSI model.

Second, we have analyzed the cardinality of the coded messages in the network and demonstrate the possibility of infinite growth in the cardinality, thus in the size of the coded messages. To rectify, we have proposed a solution based on the aging concept where each packet is given an age. Packets get older in time with each participation into a coding activity. When reaching maturity, packets are no more allowed to participate to be coded together. This process limits the life cycle of the packets and helps reducing the cardinality of the coded messages in the network.

Buffering requirements at end nodes for decoding purposes and decoding complexity are also studied for the centralized decoding approach. We have shown that the buffering size and the complexity are proportional to the number of packets exchanged during the established connection between end nodes. We have also demonstrated that the concept of aging reduces buffering requirements at end nodes.

After studying the centralized decoding, we have developed in Chapter 4 the distributed decoding which reduces the buffering and the complexity at end nodes by distributing these resource requirements over the nodes of the network. Thus with distributed decoding, buffering is required at intermediate nodes also but the overall buffering size is dramatically reduced. The basic idea behind distributed decoding is to involve the intermediate nodes of the network to engage in the decoding process by removing redundancies of delivered packets from the coded messages thus reducing their cardinality.

We have studied in Chapter 5 the loss impact and the loss recovery for both centralized and distributed decoding. We have proposed solutions for fast recovery and reliably deliver all packets exchanged in lossy networks. These solutions are based on some learning mechanism that anticipates loss detection and rapidly initiates a recovery process. The advantages of these solutions is that sending nodes are not involved in the recovery process and no acknowledgment is needed to identify lost packets.

As a conclusion, this thesis has presented a contribution for the enhancement of the NC protocols by defining a novel approach for coding, routing and reliably deliver packets in wireless linear networks while taking into account some constraints as QoS and resource requirements. The new proposed protocols can be considered as models for bandwidth management in the next generation of wireless communication systems.

Perspectives

Many interesting areas and prospects can be further investigated for future works based on this thesis:

First, the work carried out in this thesis has proven the efficiency of ACNC to enhance throughput and reliability in wireless linear networks. Further comparison with other NC protocols can be performed either to identify its superiority or detect and enhance weaknesses. Models presented in Chapter 3 can be generalized to support other possible node activities. Furthermore, the study methods applied and shown in Figures 3.4, 3.12, 4.2 and 5.3 to identify coding activities and the impact of network changes can be applied in further networking studies. The coding graph used for counting the number of transmissions an the coding opportunities can be carried out for more complex network topologies.

Second, the efficiency of ACNC in wireless linear networks can be extended to random wired and wireless networks. In this case, coding opportunities can be enhanced beyond address correlation as we did in RCNC. An example of such enhancement can be seen in multicast communication networks, where more than 2 end nodes are exchanging information. Coding at intermediate nodes can be conducted only on packets involved in a multicast session.

Third, security can be carried out as an enhancement to ACNC where coding nodes need to authenticate the originated nodes of potential address correlated packets for coding. Secured ACNC can be of benefit for cloud computing where data integrity and security are some of the major concerns when it comes to cloud computing and visualization.

Fourth, the proposed ACNC can be carried out with real implementation in various existing and emerged paradigms as AdHoc, WSN, the Internet of things (IoT), 5G and Heterogeneous Networks (HetNet). WSN and the IoT that utilize low-cost information gathering and dissemination can benefits from NC to save on throughput and energy consumption. ACNC can be used as an efficient mean of communication between sensors. To this purpose, a token that travels between pairs of nodes can be used for both gathering and dissemination.

Finally, cross layer ACNC can be carried out as throughput enhancement along the OSI layers of communication systems. Since NC can be implemented at any of the OSI

layers, an additive network coding approach where NC is applied on more than one layer is an interesting research avenue.

Bibliography

[1] S. Sankaranarayanan, P. Papadimitratos, A. Mishra, and S. Hershey, "A bandwidth sharing approach to improve licensed spectrum utilization," in *First IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks, 2005. DySPAN 2005.* IEEE, 2005, pp. 279–288.

[2] S. Katti, S. Gollakota, and D. Katabi, "Embracing wireless interference: Analog network coding," in *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 4. ACM, 2007, pp. 397–408.

[3] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, 2000.

[4] S.-Y. R. Li, R. W. Yeung, and N. Cai, "Linear network coding," *IEEE Transactions on Information Theory*, vol. 49, no. 2, pp. 371–381, 2003.

[5] R. Koetter and M. Médard, "An algebraic approach to network coding," *IEEE/ACM Transactions on Networking (TON)*, vol. 11, no. 5, pp. 782–795, 2003.

[6] C. Gkantsidis and P. R. Rodriguez, "Network coding for large scale content distribution," in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, vol. 4. IEEE, 2005, pp. 2235–2245.

[7] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft, "Xors in the air: practical wireless network coding," *IEEE/ACM Transactions on Networking (ToN)*, vol. 16, no. 3, pp. 497–510, 2008.

[8] Z. Shengli, S.-C. Liew, and P. P. Lam, "Physical layer network coding," *arXiv preprint arXiv:0704.2475*, 2007.

[9] F. Xue, C.-H. Liu, and S. Sandhu, "Mac-layer and phy-layer network coding for two-way relaying: Achievable regions and opportunistic scheduling," in *Proceedings of the 45th Annual Allerton Conference on Communication, Control and Computing*, vol. 128. Citeseer, 2007.

[10] S. Jaggi, P. Sanders, P. Chou, M. Effros, S. Egner, K. Jain, L. M. Tolhuizen *et al.*, "Polynomial time algorithms for multicast network code construction," *IEEE Transactions on Information Theory*, vol. 51, no. 6, pp. 1973–1982, 2005.

[11] T. Ho, M. Médard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *IEEE Transactions on Information Theory*, vol. 52, no. 10, pp. 4413–4430, 2006.

[12] C. Fragouli, J.-Y. Le Boudec, and J. Widmer, "Network coding: an instant primer," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 1, pp. 63–68, 2006.

[13] M. Langberg, A. Sprintson, and J. Bruck, "The encoding complexity of network coding," *IEEE/ACM Transactions on Networking (TON)*, vol. 14, no. SI, pp. 2386–2397, 2006.

[14] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, "Network flows," DTIC Document, Tech. Rep., 1988.

[15] T. Wang and G. B. Giannakis, "Complex field network coding for multiuser cooperative communications," *IEEE Journal on Selected Areas in Communications,*, vol. 26, no. 3, pp. 561–571, 2008.

[16] T. Ho, R. Koetter, M. Medard, D. R. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting," *in Proc. 2003 International Symposium on Information Theory*, 2003.

[17] J. Irvine and D. Harle, *Data communications and networks: An engineering approach.* John Wiley & Sons, 2002.

[18] J. L. Van Wyk, A. Helberg, and L. Grobler, "Comparing the implementation of network coding on different osi layers," *North West University, Potchefstroom Campus*, 2009.

[19] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek *et al.*, "Overcast: reliable multicasting with on overlay network," in *Proceedings of the 4th conference on Symposium on Operating System Design & Implementation-Volume 4.* USENIX Association, 2000, pp. 14–14.

[20] Y. Zhu, B. Li, and J. Guo, "Multicast with network coding in application-layer overlay networks," *Selected Areas in Communications, IEEE Journal on*, vol. 22, no. 1, pp. 107–120, 2004.

[21] D. Koutsonikolas, C.-C. Wang, and Y. C. Hu, "Efficient network-coding-based opportunistic routing through cumulative coded acknowledgments," vol. 19, no. 5. IEEE Press, 2011, pp. 1368–1381.

[22] J. K. Sundararajan, D. Shah, M. Médard, S. Jakubczak, M. Mitzenmacher, and J. O. Barros, "Network coding meets tcp: Theory and implementation," *Proceedings of the IEEE*, vol. 99, no. 3, pp. 490–512, 2011.

[23] C. Robinson and P. Kumar, "Sending the most recent observation is not optimal in networked control: Linear temporal coding and towards the design of a control specific transport protocol," in *Decision and Control, 2007 46th IEEE Conference on.* IEEE, 2007, pp. 334–339.

[24] S. Ahmed and S. S. Kanhere, "Vanetcode: network coding to enhance cooperative downloading in vehicular ad-hoc networks," in *Proceedings of the 2006 international conference on Wireless communications and mobile computing.* ACM, 2006, pp. 527–532.

[25] L. Scalia, F. Soldo, and M. Gerla, "Piggycode: a mac layer network coding scheme to improve tcp performance over wireless networks," in *IEEE Global Telecommunications Conference, 2007. GLOBECOM'07.* IEEE, 2007, pp. 3672–3677.

[26] S. Zhang, S. C. Liew, and P. P. Lam, "Hot topic: physical-layer network coding," in *Proceedings of the 12th annual international conference on Mobile computing and networking.* ACM, 2006, pp. 358–365.

[27] H. Rahul, H. Hassanieh, and D. Katabi, "Sourcesync: a distributed wireless architecture for exploiting sender diversity," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 171–182, 2011.

[28] S. C. Liew, S. Zhang, and L. Lu, "Physical-layer network coding: Tutorial, survey, and beyond," *Physical Communication*, vol. 6, pp. 4–42, 2013.

[29] L. Chen, T. Ho, M. Chiang, S. H. Low, and J. C. Doyle, "Congestion control for multicast flows with network coding," *IEEE Transactions on Information Theory,*, vol. 58, no. 9, pp. 5908–5921, 2012.

[30] Y. E. Sagduyu and A. Ephremides, "Cross-layer optimization of mac and network coding in wireless queueing tandem networks," *IEEE Transactions on Information Theory,*, vol. 54, no. 2, pp. 554–571, 2008.

[31] K.-H. Lee and J.-H. Kim, "Cross-layer design for tcp splitting connections with network coding in dvb-rcs networks," in *2014 International Conference on Information and Communication Technology Convergence (ICTC),*. IEEE, 2014, pp. 970–973.

[32] K. Narayanan, M. P. Wilson, and A. Sprintson, "Joint physical layer coding and network coding for bi-directional relaying," in *Proc. of Allerton Conference on Communication, Control and Computing*, 2007.

[33] H. Seferoglu, A. Markopoulou, and K. Ramakrishnan, "I 2 nc: intra-and inter-session network coding for unicast flows in wireless networks," in *INFOCOM, 2011 Proceedings IEEE.* IEEE, 2011, pp. 1035–1043.

[34] D. S. Lun, M. Médard, and R. Koetter, "Network coding for efficient wireless unicast," in *2006 International Zurich Seminar on Communications.* IEEE, 2006, pp. 74–77.

[35] D. S. Lun, M. MéDard, R. Koetter, and M. Effros, "On coding for reliable communication over packet networks," *Physical Communication*, vol. 1, no. 1, pp. 3–20, 2008.

[36] M. Ghaderi, D. Towsley, and J. Kurose, "Network coding performance for reliable multicast," in *Military Communications Conference, 2007. MILCOM 2007.* IEEE, 2007, pp. 1–7.

[37] A. G. Dimakis, P. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *IEEE Transactions on Information Theory*, vol. 56, no. 9, pp. 4539–4551, 2010.

[38] J. Widmer and J.-Y. Le Boudec, "Network coding for efficient communication in extreme networks," in *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking.* ACM, 2005, pp. 284–291.

[39] D. Wang, Q. Zhang, and J. Liu, "Partial network coding: Theory and application for continuous sensor data collection," in *IWQoS 2006. 14th IEEE International Workshop on Quality of Service, 2006.* IEEE, 2006, pp. 93–101.

[40] K. Bhattad and K. R. Narayanan, "Weakly secure network coding," *NetCod, Apr*, vol. 104, 2005.

[41] N. Cai and R. W. Yeung, "Secure network coding," in *2002 IEEE International Symposium on Information Theory, 2002. Proceedings.* IEEE, 2002, p. 323.

[42] ——, "Secure network coding on a wiretap network," *IEEE Transactions on Information Theory*, vol. 57, no. 1, pp. 424–435, 2011.

[43] Z. Yu, Y. Wei, B. Ramkumar, and Y. Guan, "An efficient signature-based scheme for securing network coding against pollution attacks," in *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE.* IEEE, 2008.

[44] S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, and M. Médard, "Resilient network coding in the presence of byzantine adversaries," in *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE.* IEEE, 2007, pp. 616–624.

[45] H. Guo, Y. Qian, K. Lu, and N. Moayeri, "The benefits of network coding over a wireless backbone," in *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE.* IEEE, 2009, pp. 1–6.

[46] L. Mitten, "Branch-and-bound methods: General formulation and properties," *Operations Research*, vol. 18, no. 1, pp. 24–34, 1970.

[47] E. Balas, S. Schmieta, and C. Wallace, "Pivot and shifta mixed integer programming heuristic," *Discrete Optimization*, vol. 1, no. 1, pp. 3–12, 2004.

[48] J. Eckstein, "Parallel branch-and-bound algorithms for general mixed integer programming on the cm-5," *SIAM Journal on Optimization*, vol. 4, no. 4, pp. 794–814, 1994.

[49] C. Fragouli, D. Lun, M. Médard, and P. Pakzad, "On feedback for network coding," in *Information Sciences and Systems, 2007. CISS'07. 41st Annual Conference on.* IEEE, 2007, pp. 248–252.

[50] T. Ohira and R. Sawatari, "Phase transition in a computer network traffic model," *Physical Review E*, vol. 58, no. 1, p. 193, 1998.

[51] D. Nguyen, T. Tran, T. Nguyen, and B. Bose, "Wireless broadcast using network coding," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 2, pp. 914–925, 2009.

[52] R. Kumar, S. Tati, F. De Mello, S. V. Krishnamurthy, and T. L. Porta, "Network coding aware rate selection in multi-rate ieee 802.11," in *Network Protocols (ICNP), 2010 18th IEEE International Conference on.* IEEE, 2010, pp. 92–102.

[53] B. Haeupler, M. Kim, and M. Médard, "Optimality of network coding with buffers," *IEEE Information Theory Workshop (ITW)*, pp. 533–537, 2011.

[54] W. Chen, K. B. Letaief, and Z. Cao, "Buffer-aware network coding for wireless networks," *IEEE/ACM Transactions on Networking*, vol. 20, no. 5, pp. 1389–1401, 2012.

[55] C. Ibars, L. Giupponi, and S. Addepalli, "Distributed multiple access and flow control for wireless network coding," *IEEE 71st Vehicular Technology Conference (VTC 2010-Spring)*, pp. 1–6, 2010.

[56] A. Agarwal and M. Charikar, "On the advantage of network coding for improving network throughput," in *Information Theory Workshop, 2004. IEEE*, 2004, pp. 247–249.

[57] M. Ghaderi, D. Towsley, and J. Kurose, "Reliability gain of network coding in lossy wireless networks," in *IEEE INFOCOM 2008. The 27th Conference on Computer Communications.* IEEE, 2008.

[58] S. Rayanchu, S. Sen, J. Wu, S. Banerjee, and S. Sengupta, "Loss-aware network coding for unicast wireless sessions: design, implementation, and performance evaluation," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 36, no. 1. ACM, 2008, pp. 85–96.

[59] D. S. Papailiopoulos, J. Luo, A. G. Dimakis, C. Huang, and J. Li, "Simple regenerating codes: Network coding for cloud storage," in *2012 IEEE Proceedings INFOCOM*. IEEE, 2012, pp. 2801–2805.

[60] J. F. Mingorance-Puga, G. Maciá-Fernández, A. Grilo, and N. M. C. Tiglao, "Efficient multimedia transmission in wireless sensor networks," in *6th EURO-NF Conference on Next Generation Internet (NGI)*. IEEE, 2010, pp. 1–8.

[61] H. Lee, Y. Ko, and D. Lee, "A hop-by-hop reliability support scheme for wireless sensor networks," in *Fourth Annual IEEE International Conference on Pervasive Computing and Communications Workshops. PerCom Workshops*. IEEE, 2006, pp. 5–pp.

[62] C. Fragouli, D. Katabi, A. Markopoulou, M. Medard, and H. Rahul, "Wireless network coding: Opportunities & challenges," in *MILCOM 2007. IEEE Military Communications Conference*. IEEE, 2007, pp. 1–8.

[63] J.-S. Park, M. Gerla, D. S. Lun, Y. Yi, and M. Medard, "Codecast: a network-coding-based ad hoc multicast protocol," *IEEE Wireless Communications*, vol. 13, no. 5, pp. 76–81, 2006.

[64] D. G. Andersen, A. C. Snoeren, and H. Balakrishnan, "Best-path vs. multi-path overlay routing," *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, pp. 91–100, October 2003.

[65] S. Biswas and R. Morris, "Opportunistic routing in multi-hop wireless networks," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 1, pp. 69–74, 2004.

[66] K. Zeng, W. Lou, and H. Zhai, "Capacity of opportunistic routing in multi-rate and multi-hop wireless networks," *Wireless Communications, IEEE Transactions on*, vol. 7, no. 12, pp. 5118–5128, 2008.

[67] S. Chachulski, M. Jennings, S. Katti, and D. Katabi, "Trading structure for randomness in wireless opportunistic routing," vol. 37, no. 4, 2007.

[68] Z. Zhong and S. Nelakuditi, "On the efficacy of opportunistic routing," *4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, 2007. SECON'07.*, pp. 441–450, June 2007.

[69] W.-L. Yeow, A. T. Hoang, and C.-K. Tham, "Minimizing delay for multicast-streaming in wireless networks with network coding," in *INFOCOM 2009, IEEE*. IEEE, 2009, pp. 190–198.

[70] E. Drinea, L. Keller, and C. Fragouli, "Real-time delay with network coding and feedback," *Physical Communication*, vol. 6, pp. 100–113, 2013.

Résumé

Le codage de réseau (CR) est une nouvelle technique reposant sur la réalisation par les nœuds du réseau des fonctions de codage et de décodage des données afin d'améliorer le débit et réduire les retards. En utilisant des algorithmes algébriques, le codage consiste à combiner ensemble les paquets transmis et le décodage consiste à restaurer ces paquets. Cette opération permet de réduire le nombre total de transmissions de paquets pour échanger les données, mais requière des traitements additionnels au niveau des nœuds. Le codage de réseau peut être appliqué au niveau de différentes couches ISO. Toutefois dans ce travail, sa mise en œuvre est effectuée au niveau de la couche réseau. Dans ce travail de thèse, nous présentons des techniques de codage de réseau s'appuyant sur de nouveaux protocoles permettant d'optimiser l'utilisation de la bande passante, d'améliorer la qualité de service et de réduire l'impact de la perte de paquets dans les réseaux à pertes. Plusieurs défis ont été relevés notamment concernant les fonctions de codage/décodage et tous les mécanismes connexes utilisés pour livrer les paquets échangés entre les nœuds. Des questions comme le cycle de vie des paquets dans le réseau, la cardinalité des messages codés, le nombre total d'octets transmis et la durée du temps de maintien des paquets ont été adressées analytiquement, en s'appuyant sur des théorèmes, qui ont été ensuite confirmés par des simulations. Dans les réseaux à pertes, les méthodes utilisées pour étudier précisément le comportement du réseau conduisent à la proposition de nouveaux mécanismes pour surmonter cette perte et réduire la charge. Dans la première partie de la thèse, un état de l'art des techniques de codage de réseaux est présenté à partir des travaux de Alshwede et al. Les différentes techniques sont détaillées mettant l'accent sur les codages linéaires et binaires. Ces techniques sont décrites en s'appuyant sur différents scénarios pour aider à comprendre les avantages et les inconvénients de chacune d'elles.

Dans la deuxième partie, un nouveau protocole basé sur la corrélation des adresses (ACNC) est présenté, et deux approches utilisant ce protocole sont introduites ; l'approche centralisée où le décodage se fait aux nœuds d'extrémités et l'approche distribuée où chaque nœud dans le réseau participe au décodage. Le décodage centralisé est élaboré en présentant d'abord ses modèles de décision et le détail du décodage aux nœuds d'extrémités. La cardinalité des messages codés reçus et les exigences de mise en mémoire tampon au niveau des nœuds d'extrémités sont étudiées et les notions d'âge et de maturité sont introduites. On montre que le décodage distribué permet de réduire la charge sur les nœuds d'extrémité ainsi que la mémoire tampon au niveau des nœuds intermédiaires. La perte et le recouvrement avec les techniques de codage de réseau sont examinés pour les deux approches proposées. Pour l'approche centralisée, deux mécanismes pour limiter l'impact de la perte sont présentés. A cet effet, le concept de fermetures et le concept des sous-ensembles couvrants sont introduits. Les recouvrements optimaux afin de trouver l'ensemble optimal de paquets à retransmettre dans le but de décoder tous les paquets reçus sont définis. Pour le décodage distribué, un nouveau mécanisme de fiabilité saut à saut est proposé tirant profit du codage de réseau et permettant de récupérer les paquets perdus sans la mise en œuvre d'un mécanisme d'acquittement.

Abstract

Network coding (NC) is a new technique in which transmitted data is encoded and decoded by the nodes of the network in order to enhance throughput and reduce delays. Using algebraic algorithms, encoding at nodes accumulates various packets in one message and decoding restores these packets. NC requires fewer transmissions to transmit all the data but more processing at the nodes. NC can be applied at any of the ISO layers. However, the focus is mainly on the network layer level.
In this work, we introduce novelties to the NC paradigm with the intent of building easy to implement NC protocols in order to improve bandwidth usage, enhance QoS and reduce the impact of losing packets in lossy networks. Several challenges are raised by this thesis concerning details in the coding and decoding processes and all the related mechanisms used to deliver packets between end nodes. Notably, questions like the life cycle of packets in coding environment, cardinality of coded messages, number of byte overhead transmissions and buffering time duration are inspected, analytically counted, supported by many theorems and then verified through simulations. By studying the packet loss problem, new theorems describing the behavior of the network in that case have been proposed and novel mechanisms to overcome this loss have been provided.
In the first part of the thesis, an overview of NC is conducted since triggered by the work of Alshwede et al. NC techniques are then detailed with the focus on linear and binary NC. These techniques are elaborated and embellished with examples extracted from different scenarios to further help understanding the advantages and disadvantages of each of these techniques.

In the second part, a new address correlated NC (ACNC) protocol is presented and two approaches using ACNC protocol are introduced, the centralized approach where decoding is conducted at end nodes and the distributed decoding approach where each node in the network participates in the decoding process. Centralized decoding is elaborated by first presenting its decision models and the detailed decoding procedure at end nodes. Moreover, the cardinality of received coded messages and the buffering requirements at end nodes are investigated and the concepts of aging and maturity are introduced. The distributed decoding approach is presented as a solution to reduce the overhead on end nodes by distributing the decoding process and buffering requirements to intermediate nodes.

Loss and recovery in NC are examined for both centralized and distributed approaches. For the centralized decoding approach, two mechanisms to limit the impact of loss are presented. To this effect, the concept of closures and covering sets are introduced and the covering set discovery is conducted on undecodable messages to find the optimized set of packets to request from the sender in order to decode all received packets. For the distributed decoding, a new hop-to-hop reliability mechanism is proposed that takes advantage of the NC itself and depicts loss without the need of an acknowledgement mechanism.

INSA RENNES

N° d'ordre :
Institut National des Sciences Appliquées de Rennes
20, Avenue des Buttes de Coësmes • CS 70839 • F-35708 Rennes Cedex 7
Tel : 02 23 23 82 00 - Fax : 02 23 23 83 96